Lecture Notes in Computer Science 1989
Edited by G. Goos, J. Hartmanis and J. van Leeuwen

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

Marco Ajmone Marsan   Andrea Bianco (Eds.)

Quality of Service in Multiservice IP Networks

International Workshop, QoS-IP 2001
Rome, Italy, January 24-26, 2001
Proceedings

Springer

Series Editors

Gerhard Goos, Karlsruhe University, Germany

Juris Hartmanis, Cornell University, NY, USA

Jan van Leeuwen, Utrecht University, The Netherlands

Volume Editors

Marco Ajmone Marsan
Andrea Bianco
Politecnico di Torino, Dipartimento di Elettronica
Corso Duca degli Abruzzi 24, 10129 Torino, Italy
E-mail: {ajmone,bianco}@polito.it

Cataloging-in-Publication Data applied for

Die Deutsche Bibliothek - CIP-Einheitsaufnahme

Quality of service in multiservice IP networks : international
workshop ; proceedings / QoS-IP 2001, Rome, Italy, January 24 - 26,
2001. Marco Ajmone Marsan ; Andrea Bianco (ed.). - Berlin ; Heidelberg ;
New York ; Barcelona ; Hong Kong ; London ; Milan ; Paris ;
Singapore ; Tokyo : Springer, 2001
(Lecture notes in computer science ; Vol. 1989)
ISBN 3-540-41512-2

CR Subject Classification (1991): C.2, D.2, H.4.3, K.6

ISSN 0302-9743
ISBN 3-540-41512-2 Springer-Verlag Berlin Heidelberg New York

Typesetting: Camera-ready by author, data conversion by Boller Mediendesign
Printed on acid-free paper SPIN: 10782028 06/3142 5 4 3 2 1 0

Preface

IP is clearly emerging as the networking paradigm for the integration of the traffic flows generated by a variety of new applications (IP telephony, multimedia multicasting, e-business, ...), whose performance requirements may be extremely different. This situation has generated a great interest in the development of techniques for the provision of quality of service (QoS) guarantees in IP networks. Two proposals have already emerged from the IETF groups IntServ and Diff-Serv, but research and experiments are continuing, in order to identify the most effective architectures and protocols. The Italian Ministry for University and Scientific Research has been funding a research program on these topics, named "Techniques for quality of service guarantees in multiservice telecommunication networks" or MQOS for short, in the years 1999 and 2000.

At the end of its activity, the MQOS program has organized in Rome (Italy) in January 2001 the *International Workshop on QoS in Multiserevice IP Networks* (QoS-IP 2001), for the presentation of high-quality recent research results on QoS in IP networks, and the dissemination of the most relevant research results obtained within the MQOS program.

This volume of the LNCS series contains the procedings of QoS-IP 2001, including 2 invited papers as well as 26 papers selected from an open call. These very high quality papers provide a clear view of the state of the art of the research in the field of quality of service provisioning in multiservice IP networks, and we hope that these proceedings will be a valuable reference for years to come.

The preparation of this volume has benefitted from the hard work of many people: the MQOS researchers, the paper authors, the reviewers, and the LNCS staff, Alfred Hofmann in particular. We wish to thank all of them for their cooperation.

November 2000

Marco Ajmone Marsan
Andrea Bianco

Organization

QoS-IP 2001 is organized by the MQOS Program at University of Rome - La Sapienza.

Program Committee

Local Organizing Committee

Sponsoring Institutions

Agilent Technologies, Italy
Alcatel, Italy
CSELT, Italy
Ericsson Lab, Italy
Infostrada, Italy
Marconi Communications, Italy
Siemens Information and Communication Networks, Italy

Table of Contents

Design and Implementation of Scalable Admission Control

Julie Schlembach[1], Anders Skoe[2], Ping Yuan[1], and Edward Knightly[1]

[1] Department of Electrical and Computer Engineering, Rice University
[2] Department of Electrical Engineering, Stanford University

Abstract. While the IntServ solution to Internet QoS can achieve a strong service model that guarantees flow throughputs and loss rates, it places excessive burdens on high-speed core routers to signal, schedule, and manage state for individual flows. Alternatively, the DiffServ solution achieves scalability via aggregate control, yet cannot ensure a particular QoS to individual flows. To simultaneously achieve scalability and a strong service model, we have designed and implemented a novel architecture and admission control algorithm termed Egress Admission Control. In our approach, the available service on a network path is passively monitored, and admission control is performed only at egress nodes, incorporating the effects of cross traffic with implicit measurements rather than with explicit signaling. In this paper, we describe our implementation of the scheme on a network of prototype routers enhanced with ingress-egress path monitoring and edge admission control. We report the results of testbed experiments and demonstrate the feasibility of an edge-based architecture for providing IntServ-like services in a scalable way.

1 Introduction

The Integrated Services (IntServ) architecture of the IETF provides a mechanism for supporting quality-of-service for real-time flows. Two important components of this architecture are admission control [2,4] and signaling [12]: the former ensures that sufficient network resources are available for each new flow, and the latter communicates such resource demands to each router along the flow's path. However, without further enhancements (e.g., aggregation [7]), the demand for high-speed core routers to process per-flow reservation requests introduces scalability and deployability limitations for this architecture.

In contrast, the Differentiated Services (DiffServ) architecture [1,6,10] achieves scalability by limiting quality-of-service functionalities to class-based priority mechanisms together with service level agreements. However, without per-flow admission control, such an approach necessarily weakens the service model as compared to IntServ; namely, individual flows are not assured of a bandwidth or loss guarantee.

To simultaneously achieve scalability and a strong service model, we have devised Egress Admission Control [3], an architecture and algorithm for scalable

M. Ajmone Marsan, A. Bianco (Eds.): QoS-IP 2001, LNCS 1989, pp. 1–15, 2001.
© Springer-Verlag Berlin Heidelberg 2001

QoS provisioning. In this scheme, admission control decisions are made solely at egress routers; per-flow state is not maintained in the network core nor in the egress router, and there is no coordination of state with core nodes or other egress nodes. Therefore, admission control and resource reservation are performed in a distributed and scalable way. As a consequence of the scalable architecture, a key challenge is how to assess the network's available resources at an egress point. Our solution is to employ continuous passive monitoring of a path (ingress-egress pair) and assess its available service by measurement-based analysis of the arrival and service times of packets on the path. In this way, an egress router holds implicit control over other paths, by ensuring that all classes on all paths maintain their desired quality-of-service levels. This implicit control factors in effects of cross traffic and prevents other egress routers from admitting flows that would exceed the available bandwidth.

Figure 1 illustrates a simplified model of egress admission control. The figure depicts how an ingress-to-egress path is modeled as a "black box" with an unknown service discipline and cross-traffic that cannot be directly measured. An important part of egress admission control is assessing the available service along this path. We will show how the abstraction of a statistical service envelope [8] provides a general framework for characterizing service, including fluctuating available resources due to varying demands of cross traffic.

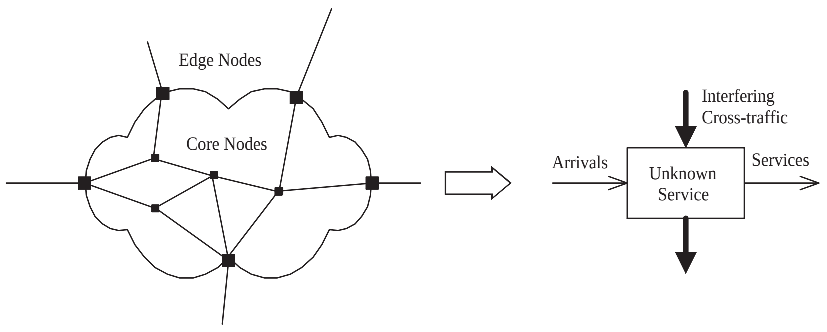


Fig. 1. Egress Admission Control System Model

This paper describes our design, implementation, and measurement study of Egress Admission Control. In particular, we illustrate the key design issues that must be addressed for scalable admission control. Compared with an IntServ architecture, our implementation addresses three issues. First, we modify RSVP so that only egress nodes process reservation requests. Second, to assess both arrivals and available service at a path's egress node, we insert timestamps and path-level sequence numbers into packets: we describe our implementation of both IPv4 time stamping and NTP clock synchronization. Finally, we implement an edge based admission control algorithm in which measurements of the service along a path are used to predict and control network-wide QoS.

To evaluate the scheme, we performed an extensive measurement study on a testbed consisting of prototype routers equipped with our implementation of Egress Admission Control. The experiments indicate that the algorithm is able to control the network's admissible region within a range such that the requested quality-of-service parameters can be satisfied. This demonstrates a key component of scalable admission control, namely, that flow-based QoS can be achieved without signaling each traversed node for each reservation request.

2 Design and Implementation

Our design consists of three inter-related components: (1) the signaling protocol by which new flows are established, (2) the traffic measurement module, including time stamping and loss detection, and (3) the admission control module which accepts or rejects requests to establish new real-time flows.

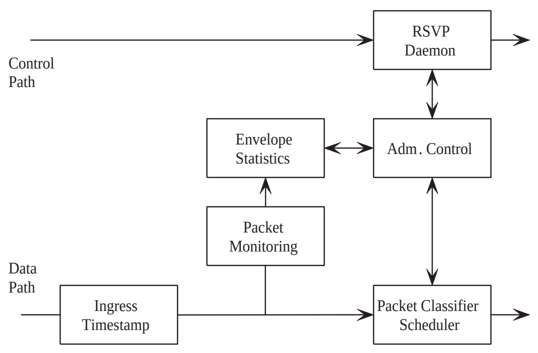


Fig. 2. Edge Router Architecture

Figure 2 depicts the relationship among these system components. As shown, when a user desires to initiate a new QoS session such as streaming video or voice over IP, the host sends a signaling message to determine if the requested service is available. The request handler then calls the admission control routine to determine whether the new flow may be admitted while satisfying all flows' statistical service guarantees. Meanwhile, the statistical properties of a path are monitored in real-time at the egress node. This information is accessed by the admission control algorithm in its decision. Each of these edge-router components communicate via shared memory.

2.1 Flow Establishment and Signaling

In order to perform flow-based admission control decisions, a signaling protocol is required to communicate a resource reservation request from a host to a domain's egress router. Consequently, we modified RSVP to establish flows in the Egress Admission Control architecture.

In general, RSVP's functionality as defined in [12] can be described as follows. When a host's application requests a specific QoS for its data stream, the host calls the RSVP daemon to deliver its request to routers along the data stream's path. Each RSVP router has several local procedures for reservation setup and enforcement. Policy control determines whether the user has administrative permission to make the reservation. Admission control keeps track of the system resources and determines whether the node has sufficient resources to supply the requested QoS. The RSVP daemon invokes both procedures before accepting the new flow. If either test fails, the RSVP daemon returns an error notification to the application that originated the request. If both checks succeed, the RSVP daemon sets parameters in the packet classifier and packet scheduler to obtain the requested QoS. The packet classifier determines the QoS class for each packet and the packet scheduler orders packet transmission to achieve the promised QoS for each flow.

In our implementation of Egress Admission Control, the requirements of RSVP are significantly simplified. Foremost, RSVP reservation messages need only be processed by a requesting flow's egress router (or each domain's egress router in the case of multiple domains). Since a prominent feature of RSVP is that it provides transparent operation through non-supporting regions, the RSVP daemon is simply not running on the core routers, and reservation requests are merely forwarded by core routers as normal IP packets. Therefore, it is only necessary to run the RSVP daemon on edge routers where admission control decisions are made.

The next step is for an edge node to identify that it is in fact the egress node and not an ingress node. This is quite simple to achieve based on the static configuration of the edge router itself: if it receives an RSVP message on its core-node interface, it is the egress node for the flow and should process the request; otherwise, the router is an ingress node and should simply forward the packet.

Our second alteration to RSVP is in the admission control algorithm. We modified the RSVP daemon to call our egress admission control program (described below) upon receipt of a new reservation request. This algorithm is invoked as an alternative to IntServ-style per-link measurement based admission control such as described in [2].

Finally, we do not send explicit tear down messages upon completion of a real-time session, as per-flow state is not maintained anywhere in the system, and all admission decisions are based on measurements. (We note however, that tear down messages may be desirable for other purposes such as resource usage accounting).

Regardless, the RSVP messages themselves are not modified. Excluding the changes made to the RSVP daemon, the protocol is compliant with the standard, and the host and user interfaces are left unaltered.

2.2 Measuring Path Traffic and Service

In order for an egress router to assess even simple characteristics of a path such as ingress-to-egress queueing delays, the egress router must measure both the egress router's service time of packets, and the times that packets entered the ingress node. Consequently, to reveal such entrance times to the egress nodes, we insert time stamps for real-time packets at ingress nodes.[1]

There are four design components to our traffic and service measurement methodology: inserting timestamps at ingress nodes, synchronizing edge-router clocks, capturing and reading timestamps along with other packet header information at the egress node, and calculating traffic envelopes from this data.

Timestamping In order to communicate packet ingress entrance times to egress routers, we insert information into fields of the IP header at the ingress node, which is ignored by the core nodes and read at the egress node. With an approach analogous to the Dynamic Packet State code [9], we utilize 18 bits from the ip_tos field and ip_off field to transmit a 10-bit timestamp as depicted in Figure 3.

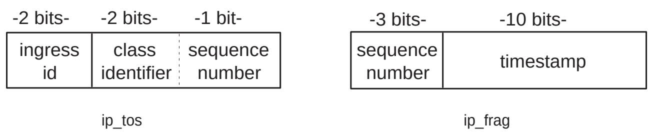


Fig. 3. IP Header Insertions for Path Monitoring

The ingress node transmits the ten most significant bits of the fraction of the second that just passed in UTC (Universal Coordinated Time) when the call to insert the arrival timestamp is summoned in the ip_input() routine. At the egress node, the service time is registered when the packet leaves the router at the outgoing interface. This ensures that when calculating the envelopes, we also account for queuing delays at both edge routers.

In our particular implementation, inserting the timestamps into the ip_tos and ip_off fields of the IP packet header ensures that the intermediate routers forward the packets without any additional processing overhead. Had the IP timestamp options been used, additional packet-processing overhead would have occurred at the intermediate nodes. This type of implementation does assume that no packets are fragmented, but this is simple to control in a laboratory setting.

[1] A modified design that would avoid time stamping would be for ingress nodes to collect arrival statistics and periodically transmit the aggregate information to the egress nodes. However, such coordination of state among edge nodes is not a part of our current design.

The aspect of the implementation illustrates the current limitations of IP options, as the existence of the IP_OPTIONS flag invokes the function ip_dooptions(). This then incurs significant overhead since the function call places the packet on the "slow path". A more efficient solution could utilize a flag indicating whether a packet should be simply be forwarded by an interior node, or whether it needs to be processed further. Such a flag would easily be detected in hardware, and hence be compatible with high-speed core routers.

NTP and Time Synchronization As explained above, the arrival time of the packet at the ingress node must be communicated in order to calculate the service envelopes at the egress node of the network. For this to occur, the clocks in the various edge routers have to be synchronized within a value that guarantees delay bounds on the order of tens of msec. We implemented the Network Time Protocol (NTP) on the ingress/egress machines to achieve this goal.

NTP operates by sending its own packets, containing the four most recent timestamps between the two host computers, at polling intervals between 64 and 1024 seconds, depending on the stability of the two clocks. The roundtrip propagation time and offset are calculated from these four timestamps. Then the computer specified to adjust its time does so by gradually skewing its clock to the "correct time" of the other clock using a phase lock loop clock discipline algorithm described in detail in [5]. This algorithm alters the computer clock time, while compensating for the intrinsic frequency error and dynamically adjusting the poll interval. The measured time errors discipline a feedback loop, which controls the phase and frequency of the clock oscillator. This is done with the aid of the clock adjust process, which runs at intervals of one second.

For our testbed described in Section 3 and moderate workloads, the offset between the edge routers' clocks was typically 0.2 msec, whereas the offset value increased as high as 2 msec when the router was under a heavy load. Regardless, as queuing delays and transmission delays are significantly larger than this offset, the timestamps are accurate enough to calculate the path characteristics at the egress router using the timestamp information from the ingress node.

An important feature of NTP relevant for this research is its scalability: its phase lock loop clock discipline is designed for large-scale networks so that NTP can maintain low offsets among clocks, even if propagation delays increase significantly [5].

Regardless, our future plan is to employ Global Positioning System receivers for clock synchronization among edge routers. The key problem of NTP is that under heavy load, constantly fluctuating offsets may cause frequent clock readjustments by NTP. During each readjustment, packets transmitted at a later time may have smaller time stamps than packets transmitted earlier. Such errors in turn interfere with path monitoring.

Capturing Traffic at Egress Nodes In order to collect timestamps of all real-time packets traversing a path, we modified the packet sniffer tcpdump[2], which

[2] www.tcpdump.org

uses the libpcap library to read header information from IP and TCP/UDP headers. While tcpdump is not the ideal way to perform this task, since it is performed at user level, it is sufficient for the 10 Mb/sec routers employed in the testbed. For high-speed implementation, this functionality could be integrated into the kernel or supplemented with hardware support.

In addition to timestamps, the egress router records the packet length, an ingress node identifier, a class identifier, and a four-bit per-path sequence number (as depicted in Figure 3). The class and ingress node identifier will allow admission control on a per-class, per-path (ingress-egress pair) basis.

The per-path sequence number is used to identify loss on the path at the egress point. Using a 4-bit sequence number ensures that up to 15 consecutive packet losses can be detected. In our current implementation, we have not used this loss detection, as the class QoS requirements are stringent enough to keep loss sufficiently low that dropped packets can be ignored. However, for less stringent QoS requirements, such loss would need to be incorporated: otherwise egress routers may over-estimate the available service along a path, as not all arriving traffic is incorporated into the measurement.

Computing a Path's Available Service To assess the available service on a path, we measure a path's statistical service envelope [8], a general characterization of the end-to-end service received by a traffic class. This service abstraction can incorporate the effects of interfering cross traffic without explicitly measuring or controlling it. Moreover, the service envelope exploits features of the backbone nodes' schedulers and the effects of statistical resource sharing at both the flow level and the class level. For example, if a class is provided a circuit-like service without sharing among traffic classes, the service envelope will measure a simple linear function. In contrast, if the network performs scheduling similar to weighted fair queuing, the service envelope will reflect the available capacity beyond the minimum "guaranteed rate" which can be exploited by the class, i.e., the excess capacity which is available due to fluctuating resource demands of cross traffic and other traffic classes. Finally, by limiting a class' traffic through controlling admission of flows into the class, we can ensure that the class' predicted quality-of-service is within its requirements. When all edge routers perform the algorithm, the scheme ensures that all classes of all paths receive their desired service levels.

There are two methods for calculating the traffic envelope. One approach is to use the method where a rate-based envelope is considered; that is, calculate the peak rate for a given interval length. A second, and analogous method, is to calculate the total number of bytes that arrived in an interval - that is, measure the minimum interval length over which a certain number of bytes is transmitted. In our implementation, we use the latter approach, as it removes a number of divisions by the interval size from the algorithm.

In other words, instead of determining the maximum number of bytes to arrive in an interval of given size (via a sliding window) or discretizing the time scale, we calculate the minimum time required for a certain number of bytes to arrive/ be serviced or discretize the scale for the number of bytes that arrived. The primary motivation for this design decision is the fact that the flow requests specify a statistical delay bound, and following the latter scheme ensures that the variance remains in the time-domain. Consequently, the admission control equation (described below) can be applied directly without undergoing computationally expensive conversions.

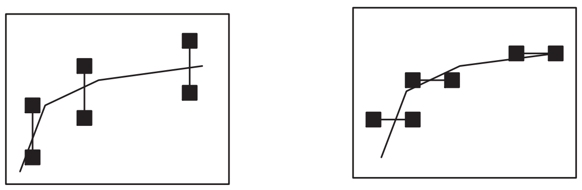

(a) Interval Discretization (b) Data Discretization

Fig. 4. Envelope Illustrations of Bytes vs. Interval Length

Admission Control An egress router's admission control decision is described as follows and is illustrated in Figure 5 (see [3] for further details). Consider a system where a traffic class between a particular ingress-egress pair has a measured peak rate arrival envelope with mean $\bar{R}(t)$ and variance $\sigma^2(t)$. In other words, over successive measurement windows, the average maximum number of arrivals is given by $t\bar{R}(t)$, and its variance is given by $t^2\sigma^2(t)$. Similarly, the class' minimum service envelope has measured average $\bar{S}(t)$ and variance $\Psi^2(t)$. The new flow with peak rate P is admissible with delay bound D if

$$t\bar{R}(t) + Pt - \bar{S}(t+D) + \alpha\sqrt{t^2\sigma^2(t) + \Psi^2(t+D)} < 0$$

where α is set according to the required violation probability [3]. Moreover, we ensure the stability condition that

$$\lim_{t\to\infty} \bar{R}(t) < \frac{\bar{S}(t)}{t}.$$

Notice that for a first-come-first-serve server with link capacity C, $\frac{\bar{S}(t)}{t} = C$. For a given threshold, the algorithm determines whether the network can satisfy the new flow's quality of service requirements. If both of the above requirements are satisfied, a message to admit the new flow is relayed by the request handler and the user may begin sending the traffic.

This approach pushes the task of network resource management to the edge of the system, and does not require that interior nodes perform per-flow or per-class bandwidth reservation. Instead, edge nodes exclusively handle admission control decisions and signaling messages. The available service in the contiguous interior of the network is inferred by inserting timestamps into packet headers at the ingress nodes and statistically analyzing the path's characteristics.

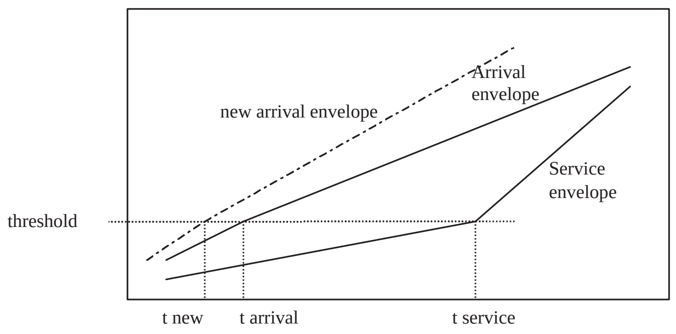


Fig. 5. Illustration of Admission Control

3 Experimental Design and Measurements

In this section, we present the results of a measurement study obtained using our implementation of egress admission control in a network of prototype routers.

3.1 Scenario

In order to study the performance of the scheme, we perform four different experiments, each building upon the previous one. All routers and hosts run the FreeBSD v3.2 operating system and are connected via 10 Mb/sec links. The buffer size of the routers is 250 packets, which for a link capacity of 10 Mb/sec and MTU of 1500 bytes corresponds to a maximum queuing delay of 300 milliseconds. We begin with a baseline experiment depicted in Figure 6, in which a single node functions as both the ingress and egress router. This router resides between the source machine, which generates the traffic, and the destination machine, which receives the traffic. In all cases, each host will generate multiple flows, and hosts are connected to routers via 100 Mb/sec links so that no queueing occurs in hosts.

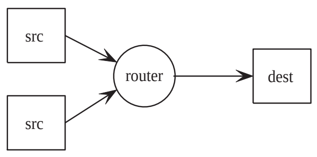


Fig. 6. Baseline Experiment

The remaining experiments are performed with the configuration depicted in Figure 7. Here, three routers interconnect five hosts. Depending on the experiment, these routers function as ingress, egress and/or core routers. Experimental results are reported between the hosts labeled “src” and “dest” whereas the two hosts labeled “cross traffic src” 1 and 2 function as cross-traffic generators to congest routers and test the egress admission control algorithm’s ability to infer the available service along a path with unmeasured cross traffic.

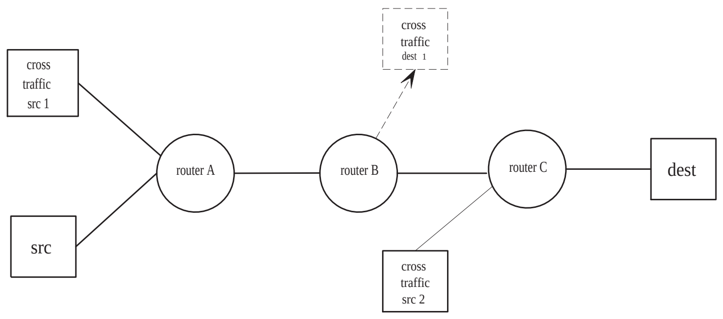


Fig. 7. Ingress/Egress Pair Experiments

3.2 Traffic Generation

To emulate the behavior of realistic real-time flows, we designed a Pareto on-off traffic generator that transmits packets only after the QoS request is admitted by the egress router’s admission control test. It consists of two components: the actual traffic generation and communication with RSVP. Packets are generated according to the Pareto on-off model with the following parameters: packet size 1000 bytes, mean burst time 250 msec, mean idle time 250 msec, and peak rate 400 kb/sec. The Pareto shape parameter is 1.9 (recall that a Pareto shape parameter less than 1 results in an infinite mean while a shape parameter less than 2 results in an infinite variance) and the flow lifetime is 5 minutes. Thus, this traffic generator produces highly bursty traffic which, when aggregated, forms a flow that exhibits self-similarity [11].

The second part of the traffic generator handles communication with RSVP. As RSVP is a receiver oriented reservation protocol, it needs both the path information from the source host and the QoS request information from the destination host. In order to communicate the user’s request to the egress router, a module in the traffic generation program on the source host side generates a path message, while a module on the destination side sends a reserve message with the QoS request. By calling the RSVP application interface function, the sender side receives the admission control result.

3.3 Measurements

Experiment 1: One Node The first experiment consists of a single router, one class, and no cross traffic. Pareto on-off traffic is sent from the main source host to the router, and then to the destination host as depicted in Figure 6. The peak rate of each flow is 800 kb/sec and the mean rate is 400 kb/sec. The link capacity is manually configured to be 9 Mb/sec using ATLQ. Thus, under a peak rate allocation scheme, 11 flows would be admitted, and to ensure stability, no more than 22 flows can be admitted. In this simplified scenario, the service envelope is simply S(t) = 9t, and we configured S(t) manually (rather than measuring it) in order to establish a performance benchmark in the case in which service is known and largely deterministic. Moreover, with the single router configuration, timestamps are not required as the router monitors the original packet entrance times (similar to the case of IntServ MBAC [2]). For the experiments, α, the parameter which controls the fraction of packets violating the class delay bound (and hence controlling the loss probability as some of these packets will be dropped when the buffer is full) is 1.0. The arrival envelope is computed by evaluating the output of tcpdump at the end of each one-second interval.

Delay Reqst (msec)	Number of Flows	Mean Delay (msec)	Maximum Delay (msec)	% Outside Bound
5	16.0	1.52	17.9	1.25
10	16.3	1.83	22.5	1.20
20	18.0	2.35	36.8	0.56
60	21.1	12.85	124.7	6.16

Table 1. Single Node Baseline Experiments

We make the following observations about the experimental results reported in Table 1.[3] First, the algorithm has exploited statistical multiplexing gains, even in this scenario of a moderate number of traffic flows. The average link utilizations are in the range of 67% to 94%. As a consequence of overbooking, violations of the target delay occur and the fifth column indicates that the violations range from 0.56% to 6.16% of packets. Second, observe that assigning different delay targets has the desired impact on measured performance, allowing mean delays in the range of 1.52 msec to 12.85 msec, and maximum delays in the range of 17.9 msec to 124.7 msec. Hence, the algorithm provides the basic mechanisms for performance differentiation in multi-class networks. Finally, we observe that the targeted violations due to statistical multiplexing are not precisely met, as the percentage of violations differs in the four cases despite having the same α of 1.0 for all experiments. The differences arise from a number of factors: the quantization of the measured arrival process; the discrete nature of flows themselves (a discrete number of flows is admitted, whereas to achieve the precise QoS target may require between N and $N+1$ flows); the strong impact on QoS for each new flow in the regime of a moderate number of flows; and the extreme burstiness of the traffic itself.

[3] All reported measurements refer to average results from at least three experiments.

Experiment 2: Ingress-Egress Pair In the second set of experiments, we consider multiple routers but without cross traffic. As depicted in Figure 7, the system contains an ingress, core, and egress router (A, B, and C respectively), and traffic is transmitted between the two hosts at ingress node A and a single destination host attached to egress router C.

Here, we establish Pareto on-off flows with peak rate 400 kb/sec, mean rate 200 kb/sec, and a link capacity of 9 Mb/sec, so that the range of admissible flows is 22 to 44. The target delay bound is 20 msec and $\alpha = 0.6$.

Figure 8 displays an example arrival and service envelope used to make an admission decision at a particular time instance of the experiment. Observe that the arrival and service envelopes have crossed indicating that the stability condition is satisfied. Moreover, observe the general concavity of the arrival envelope and convexity of the service envelope. Convexity of service envelopes is normally evident in multi-class scenarios, for if a flow remains continually backlogged over longer interval lengths, it attains a greater service on average, due to the fluctuating demands of other flows in other classes. In this case with a single class and no cross traffic, one may expect the service to be closer to linear, i.e., S(t) = 9t. However, there are two reasons for its convexity. First, other traffic is still traversing the links, including RSVP messages, NTP traffic, and other minor but noticeable background traffic such as NFS traffic. Second, the empirical service envelope is actually an approximation to the true available service. For example, while an infinite rate input flow would indeed measure a service envelope of 9t, a "minimally backlogged" flow, such as described in [8] would measure a lower service envelope due to its own rate variations.

In the experiments, the maximum number of admitted flows is 38, corresponding to an average link utilization of 84%, quite similar to utilizations obtained in theory for similar types of flows (see [4] for example).[4] In this scenario, we measured a mean delay of 3.77 msec and a maximum delay of 21.6 msec, with the percentage of packets exceeding the delay bound of 20 msec measured to be 0.0045%. (Refer to Table 2 below for summary results.)

Experiment 3: Cross Traffic- Congested Ingress Router In these experiments, we introduce cross traffic between the host labeled "cross traffic src 1" and "cross traffic dest 1." Thus, the cross traffic flows traverse routers A and B whereas the test flows traverse routers A, B, and C. Consequently, egress router C has no explicit measurements of the cross traffic, and must rely on its own path measurements to assess the available capacity on the link between routers A and B. We establish 22 on-off flows as described above for cross traffic, which correspondingly reduces the available capacity along path A-B-C. As in Experiment 2, the delay request for users' traffic is 20 msec and α in the algorithm is again set to 0.6. With the 22 cross-traffic flows, 12.7 of the A-B-C source's flows were admitted on average. Thus, the egress node has inferred the reduction in available service as compared to experiment 3 and significantly reduced its

[4] Unfortunately, no precise theoretical multi-node admission control algorithm yet exists for comparison.

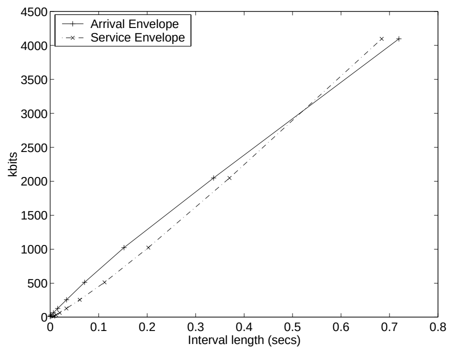


Fig. 8. Measured Arrival and Service Envelopes

number of admitted flows. Regardless, the 12.7 admitted flows correspond to 34.7 flows on link A-B, three less than allowed in Experiment 2, indicating that the un-measured cross traffic has caused the algorithm to slightly under admit. In the experiments, the mean packet delay is 3.58 msec, and the maximum delay is 18.1 msec, which led to 0% of the packets exceeding the requested target.

Experiment 4: Cross Traffic- Congested Egress Router For the final experiments, we establish cross traffic sessions through the egress router rather than the ingress router. While node C is the egress point for both the A-B-C flows and the cross traffic, these flows do not share the same path, and hence are treated separately by node C. Thus, egress router C must again implicitly discover the cross traffic's effect on the available service. Again, 22 flows of the cross traffic were established, with α set to 0.6 and a delay request of 20 msec.

In the experiments, 13.7 flows from the primary host on path A-B-C were admitted by the egress router, totaling 35.7 flows when combined with the cross traffic. This closely approximates the 83.7% utilization achieved in Experiment 2. In the experiments, the mean packet delay is 2.57 msec, the maximum delay is 21.2 msec, and the percentage of packets exceeding 20 msec delay is 0.0066%.

Exp. No.	Number of Flows	Mean Util.	Mean Delay	Maximum Delay	% Outside Bound
2	37.7	0.837	3.77 msec	21.6 msec	$4.5 \cdot 10^{-3}$
3	22+12.7	0.770	3.58 msec	18.1 msec	0
4	22+13.7	0.793	2.57 msec	21.2 msec	$6.6 \cdot 10^{-3}$

Table 2. Multiple Router Experiments

Comparing experiments 2, 3, and 4, variations in the total number of admitted flows should be expected, as the scenarios have different queueing and multiplexing characteristics. However, the percentage of packets outside the targeted delay bound would ideally be nearly identical in all three cases, at least to within the granularity of a traffic flow. Figure 9 further illustrates performance differences in the three experiments by depicting the different delay histograms in each case. While these measurements illustrate the difficulties of achieving precise control of end-to-end quality-of-service measures, the experiments do indicate that the algorithm can control admissions so that empirical quality-of-service have a strong correspondence to the targeted values.

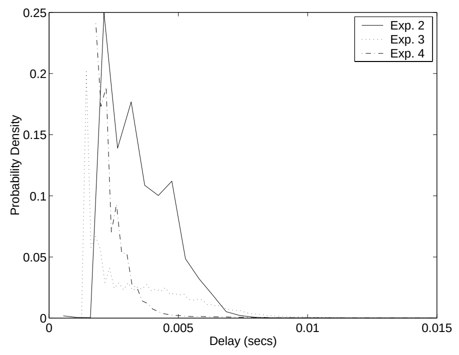


Fig. 9. Delay Distribution

4 Conclusions

This paper describes our design, implementation, and measurements of Egress Admission Control, an architecture and algorithm designed to combine the strong service model of IntServ with the scalability of DiffServ, without sacrificing network utilization. While some aspects of scalability cannot be explored in a laboratory setting, our results demonstrate a key component of scalable admission control, namely, that admission control, signaling, and state management, need not be performed at each node traversed by a flow. Instead, with proper monitoring and control of the available service on an ingress-egress path, network wide quality-of-service can be ensured while signaling only egress nodes.

Acknowledgements

The authors are grateful to Dayong Huang for designing the envelope visualization tools, and to members of the Rice Networks Group for the helpful discussions. We would also like to thank Ion Stoica for the use of his DPS code and Brent Hendricks for sharing his knowledge of FreeBSD.

References

1. S. Blake et al. An architecture for differentiated services, 1998. Internet RFC 2475.
2. L. Breslau, S. Jamin, and S. Shenker. Comments on the performance of measurement-based admission control algorithms. In *Proceedings of IEEE INFOCOM 2000*, Tel Aviv, Israel, March 2000.
3. C. Cetinkaya and E. Knightly. Scalable services via egress admission control. In *Proceedings of IEEE INFOCOM 2000*, Tel Aviv, Israel, March 2000.
4. E. Knightly and N. Shroff. Admission control for statistical QoS: Theory and practice. *IEEE Network*, 13(2):20–29, March 1999.
5. D. Mills. On the accuracy and stability of clocks synchronized by the Network Time Protocol in Internet systems. *Computer Communications Review*, 20(1), January 1990.
6. K. Nichols, V. Jacobson, and L. Zhang. Two-bit differentiated services architecture for the Internet, 1999. Internet RFC 2638.
7. P. Pan, E. Hahne, and H. Schulzrinne. BGRP: A framework for scalable resource reservation, 2000. Internet Draft, draft-pan-bgrp-framework-00.txt.
8. J. Qiu and E. Knightly. Inter-class resource sharing using statistical service envelopes. In *Proceedings of IEEE INFOCOM '99*, New York, NY, March 1999.
9. I. Stoica and H. Zhang. Providing guaranteed services without per flow management. In *Proceedings of ACM SIGCOMM '99*, Cambridge, MA, August 1999.
10. B. Teitelbaum et al. Internet2 QBone: Building a testbed for differentiated services. *IEEE Network*, 13(5):8–17, September 1999.
11. W. Willinger, M. Taqqu, R. Sherman, and D. Wilson. Self-similarity through high-variability: Statistical analyisis of Ethernet LAN traffic at the source level. *IEEE/ACM Transactions on Networking*, 5(1):71–86, February 1997.
12. L. Zhang, S. Deering, D. Estrin, S. Shenker, and D. Zappala. RSVP: A New Resource ReSerVation Protocol. *IEEE Network*, 7(5):8–18, September 1993.

Analysis and Performance Evaluation of a Connection Admission Control Scheme Based on the Many Sources Asymptotic

Giulia Bernardini, Stefano Giordano, Gregorio Procissi, and Sandra Tartarelli

Department of Information Engineering
University of Pisa

Abstract. A parsimonious traffic characterisation allows the design of efficient measurement based *Connection Admission Control (CAC)* algorithms. In recent years the notion of *effective bandwidth* (EB) has been successfully employed to quantify the amount of bandwidth to allot a connection in order to meet its Quality of Service (QoS) requirements. The EB function depends on two parameters, namely the *time scale* and the *space scale*, whose values represent the link operating point and they are related to the link capacity, the buffer size and the traffic mix. In this paper we present a study of the Many Sources Asymptotic (MSA), a Large Deviations technique that employs the notion of EB to evaluate the performance of a queueing system. Since the MSA requires to determine the time and space parameters, we firstly analysed their sensitivity to the variation of the traffic mix. We subsequently applied the results of this analysis to the refinement of a CAC algorithm based on the MSA, by using suitable thresholds for bounding the smallest mix variation beyond which a new estimation of the link operating point is required. Finally, we compared the performance of a CAC algorithm employing first the MSA then the *Large Buffer Asymptotic* (LBA) to estimate the bandwidth requirement of the active calls.

1 Introduction

The notion of effective bandwidth can significantly simplify CAC algorithms. The Large Deviations theory provides in fact results that relate the EB to the performance of a queueing system. Specifically it is possible to quantify the amount of bandwidth required by a connection in order to satisfy its QoS constraints in terms of buffer overflow probability. The EB function depends on two parameters, namely the *time scale* and the *space scale* [11], whose values represent the link operating point and they are related to the link capacity, the buffer size and the traffic mix. In the recent years two distinct types of analysis have been developed, that involve the concept of effective bandwidth, namely the *Many Sources Asymptotic* (MSA) and the *Large Buffer Asymptotic* (LBA). Both techniques provide asymptotics for the buffer overflow probability. The former analysis assumes that the number of multiplexed sources tends to infinity, whereas the latter is valid for very large buffers. In this paper we intend to compare the performance of the same measurement based CAC algorithm, that employs first the MSA and then the LBA to estimate the bandwidth requirement of the active calls. The choice of a measurement based scheme [10] is motivated by the fact that the analytical evaluation of the effective bandwidth of a traffic stream requires the full characterisation of the underlying process. Therefore its application in a practical environment is not trivial. However, the effective bandwidth can be successfully estimated based on traffic measurements (see [9] and references therein). We simulated several operational conditions to test the efficiency of the admission control system under the two different methods. Our study shows that the CAC scheme based on the MSA allows to achieve the best performance as compared to the LBA method. However, the MSA-based technique has the drawback of being computationally more complex. Indeed it requires to evaluate a new operating-point every time there is a variation in the traffic mix. Motivated by this remark, in this paper we focus on the feasibility of placing proper thresholds in the traffic mix composition. Only after one of these thresholds is crossed, the system triggers the computation of a new working-point. In order to determine the values of such thresholds we first investigate the sensitivity of the working-point to variations of the traffic mix. This paper is organised as follows. In section 2 we review the theory that leads to the MSA and LBA methods. In section 3 we subsequently investigate the sensitivity of the time and space scale parameters with respect to variations in the mix composition under different load conditions. We then apply the MSA and LBA theories to implement an admission control scheme in section 4, where we also test its effectiveness by running a set of simulations. Finally, we propose to improve the computational efficiency of the MSA-scheme by introducing proper thresholds to reduce the number of updates of the system operating-point.

2 Large Deviations Asymptotic

In the first part of this section we recall the main results on the asymptotic for the queue length of a multiplexer loaded by a large number of traffic sources and served at a constant (and independent of the number of sources) rate. The analysis, commonly known as Many Sources Asymptotic, involves the notion of shape function and basically follows the approach from Botvich *et al.* [1,2]. In the second part we present a different approach, namely the Large Buffer Asymptotic. The latter deals with a multiplexer where we let the buffer size be large enough (instead of the number of sources): this technique leads to the so-called Effective Bandwidth approximation.

2.1 The Many Sources Asymptotic

Let us consider a multiplexer with N input sources, served at a fixed rate C (independent of N) and with large buffer size B. Denote further with Q the queue-length, with A_t the total amount of work arriving at the system in the last t times and with $W_t = A_t - Ct$ the workload process. Finally let b and c be respectively the buffer size and the service rate scaled to the number of sources N, i.e. $B = bN$ and $C = cN$. Under fairly general conditions (see [1,2]) it can be shown that the following holds:

$$\lim_{N\to\infty} \frac{1}{N} \log Pr\{Q > bN\} = -I(b) \tag{1}$$

where $I(b)$ is the *shape function* defined as

$$I(b) = \inf_{t>0} t\lambda_t^* \left(\frac{b}{t}\right) \tag{2}$$

the * operator denotes the Legendre-Fenchel transform:

$$f^*(x) = \sup_\theta \{\theta x - f(\theta)\} \tag{3}$$

and λ_t is the *finite time cumulant generating function* of the workload process W_t.

$$\lambda_t(\theta) = \frac{1}{t} \log Ee^{\theta W_t} \tag{4}$$

Roughly speaking, result (1) tells that, in a multiplexer fed by a large number of sources, the probability that the queue-length exceeds a given threshold B can be approximated with:

$$P(Q > B) \approx e^{-NI(B/N)} \tag{5}$$

Of course, expression (5) can be used as long as the shape function I is known. This typically happens when:

– either the source models are known so that I can be analytically computed
– or the source models are unknown but I can be estimated based on traffic measurements.

If we assume that sources are classified into J different classes and n_j is the portion of type j sources, that is $n_j = N_j/N$ (where N_j is the number of type j sources, with $1 \leq j \leq J$ and N is the total number of sources), fairly easy calculations show that (1) is equivalent to the well known result from Courcoubetis *et al.* ([3]), obtained for a discrete time system:

$$\lim_{N\to\infty} \frac{1}{N} \log P\{Q > Nb\} = \sup_t \inf_\theta \left\{ \theta t \sum_{j=1}^{J} n_j \alpha_j(\theta, t) - \theta(b + ct) \right\} \tag{6}$$

where $\alpha_j(\theta, t)$ is the *effective bandwidth* of the type j sources:

$$\alpha_j(\theta, t) = \frac{1}{\theta t} \log Ee^{\theta A_{j,t}} \tag{7}$$

Since expressions (6)-(7) are widely used throughout this paper, it is worth pointing out some of the most relevant aspects they involve.

Firstly, notice that (6) and (7) contain the parameters t and θ, called *time scale* and *space scale* respectively. In particular, the probability in (6) depends on the value the right hand member assumes for a particular pair (θ^*, t^*), the one for which the sup-inf is attained. Such a pair represents the working-point of the system and basically depends on traffic characteristics and resource availability (i.e. service rate and buffer size). Roughly, the time scale parameter corresponds to the most likely duration of the busy period preceding an overflow event, while the space scale is related to the sources multiplexing level and basically depends on the ratio between their peak-rate and the link capacity. To give a more"physical" interpretation to both parameters (see also [4,5,11]), by denoting with $\gamma = -\log(P_{over})$ and differentiating against the buffer size and the link capacity, we obtain:

$$\theta = \frac{\delta\gamma}{\delta B} \text{ and } \theta t = \frac{\delta\gamma}{\delta C} \tag{8}$$

In other words, θ represents the rate at which the logarithm of the overflow probability decays when the buffer size increases with fixed service rate C while the product θt expresses the decay rate of the logarithm of the overflow probability when the link capacity increases, for fixed buffer size B. Finally, evaluating (7) in the pair (θ^*, t^*), we get the effective bandwidth of each source that is their need of bandwidth in the considered working conditions.

2.2 The Large Buffer Asymptotic

A different asymptotic for the queue-length based on Large Deviations (see for example [6,7]) analysis can be obtained by letting the threshold B be large enough, regardless of the number of sources. In particular, with the usual notation, under fairly general conditions, the following holds:

$$\lim_{b\to\infty} \frac{1}{B} \log P\{Q > B\} = -\inf_x \frac{I(x+C)}{x} = -\delta(C) \tag{9}$$

where now, $I(x)$ is called rate-function and plays a role very similar to that of the shape function. $I(x)$ is defined as:

$$I(x) = \lambda^*(x) = \sup_\theta\{\theta x - \lambda(\theta)\} \tag{10}$$

with $\lambda(\theta)$ the scaled cumulant generating function of the workload process W_t:

$$\lambda(\theta) = \lim_{t\to\infty} \lambda_t(\theta) = \lim_{t\to\infty} \frac{1}{t} \log E e^{\theta A_t} \tag{11}$$

Beyond the mathematical expressions, the previous result means that for multiplexer with large buffers, the asymptotic buffer overflow probability goes to zero versus the buffer size as slow as:

$$P\{Q > B\} \approx e^{-\delta(C)B} \tag{12}$$

The expression (12) is known in literature as *effective bandwidth approximation.* If we now require that the overflow probability does not exceed a given value Γ, by solving (12) with respect to C, we can get the minimum service rate a source has to be served with in order to meet its QoS level, obtaining its effective bandwidth as:

$$C_{eff} = \left.\frac{\lambda(\theta)}{\theta}\right|_{\theta=-\frac{\log \Gamma}{B}} \tag{13}$$

It is worth noticing that, although in this case the definition of effective bandwidth is slightly different from the one previously introduced for the MSA asymptotic, the expressions (13) and (7) are closely related. Furthermore, we can still identify a working-point, which in this case is defined by $\theta^* = -\frac{\log \Gamma}{B}$ and thus it does not depend on the traffic mix.

3 Time Scale and Space Scale Parameters Sensitivity

In section 2.2 we showed that, according to the many sources asymptotic, the probability that the queue occupancy crosses a threshold B depends on the working-point of the system, identified by the pair (θ^*, t^*). In this section we report our analysis of the sensitivity of the time and space parameters to traffic mix variations. This study aimed at investigating how tight is the bond between the values of the pair (θ^*, t^*), derived from equation (6), and the multiplexed sources mix. If the link utilisation is low, we expect the working-point to be scarcely influenced by small variations in the number of sources. The link is under-loaded, therefore there is no resource contention, independent of the degree of multiplexing. On the contrary, if the link utilisation is high, the sensitivity should grow and also the statistical multiplexing should have a major effect on the result. The knowledge of the parameters sensitivity to mix variations can help to speed up CAC algorithms based on MSA, as we will discuss later in the paper.

Our analysis involves three (J=3) different types of sources:

- voice sources, modelled by an on-off Markov chain with peak rate 64Kbps and average time spent in the "on" and "off" states equal to 352msec and 650msec respectively;
- data sources, namely a Bellcore Ethernet trace;
- video sources, represented by an MPEG sequence of the movie "Asterix".

Table 1 shows mean value, variance and Hurst parameter for each source.
We considered a link capacity C =155Mbps, a buffer size B =100 cells, a time unit TU=1msec and the following realistic hypotheses:

- high degree of multiplexing (N $\simeq$ 1000);
- a realistic traffic mix, i.e. the number of voice sources prevails over data and video sources and video traffic is bounded in bandwidth.

	Mean value (Cell/TU)	Variance (cell^2/TU^{2H})	Hurst parameter
Voice	0.0529094	0.00518678	0.5
Data	0.855524	0.76352	0.76
Video	1.22682	1.41854	0.77

Table 1.

Since we employed three types of sources, in each simulation we let the number of one traffic source increase, while keeping constant the remaining mix. This means that the link utilisation ρ slightly changes within each trial. We consequently classified the simulation scenarios according to the link utilisation, as follows:

- $\rho < 0.65$ *low utilisation*
- $0.65 < \rho < 0.8$ *middle utilisation*
- $\rho > 0.8$ *high utilisation.*

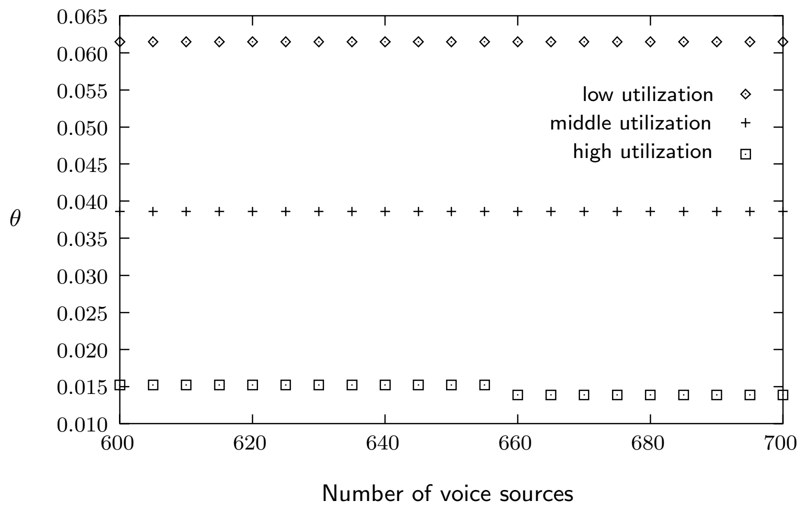


Fig. 1. Variation of the parameter θ for voice traffic

We studied separately the behaviour of the three types of sources: voice, video and data.

In figure 1 we reported the results of the sensitivity of the θ parameter for voice flows. We repeated the experiment for three different utilisation levels. In each case we let the number of voice sources vary between 600 and 700, and we changed the number of video and data sources, so that the total utilisation was different in the three cases. Partly due to the low mean value and partly to the statistical characteristics of the on-off processes, we observe that even a variation of 100 sources has little impact on the θ parameter. What is instead relevant is the utilisation level of the link. As for the time parameter t, it never changed in all these experiments and amounted to 10.

A similar test was performed for the video traces, as illustrated in figure 2. For the number of video sources, we considered a much narrower range, between 20 and 30. We observe that for video flows the sensitivity of the θ parameter is very high, due to both the higher mean value and the bursty nature of video traffic. On the contrary, the time parameter t remains also in this case always equal to 10.

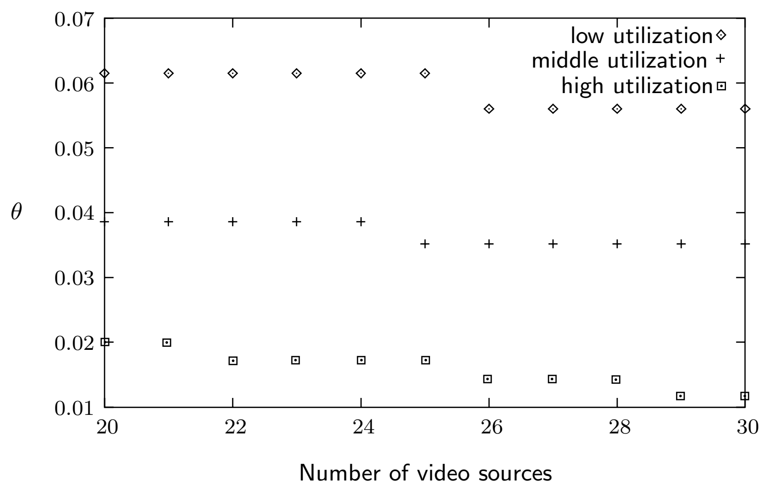


Fig. 2. Variation of the parameter θ for video traffic

Finally, we performed a slightly different test for data sources. As illustrated in figure 3, we let the number of data sources vary in a very wide range, between 300 and 550. We kept the number of voice and video sources constant in the regions of the figure indicated as low and middle link utilisation, while we increased the number of video sources (from 20 to 30) to obtain a high link utilisation. The variability of the θ parameter has an intermediate behaviour, as compared to the previous two cases. The time parameter t remains in this case equal to 10 until the number of data sources reaches 530. For a greater number of data flows, t results equal to 20. The link utilisation is in this range bigger than 0.94.

Based on the above results, we conclude that:

– The time parameter is less sensitive to traffic mix variations than the space parameter. This can be explained by recalling what already stated in section

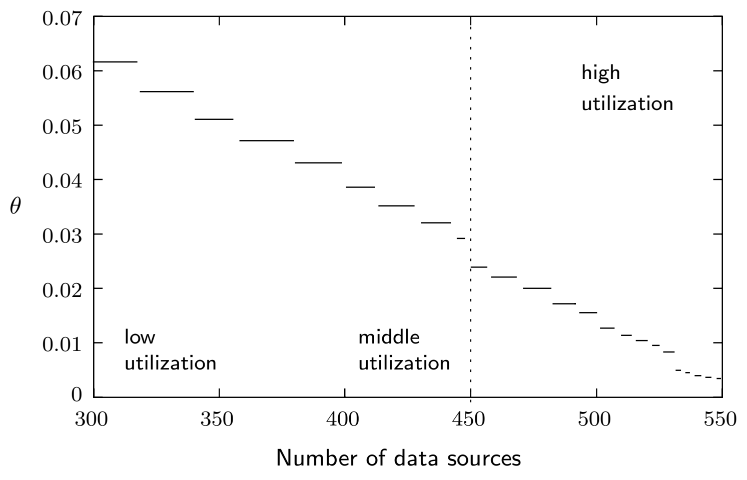


Fig. 3. Variation of the parameter θ for data traffic

2.1, i.e. that t corresponds to the most probable duration of the busy period prior to the overflow and it depends on the ratio between the buffer size B and the quantity $C - \lambda_T$, where λ_T is the mean value of the multiplexed sources. Therefore minor changes in the latter do not significantly affect t^*.

- The parameter sensitivity grows significantly with the link utilisation.
- The range of mix variation, within which the two parameters can be considered constant, depends on the type of source that provokes the variation: a larger variation in the number of data and voice sources has to occur in order to cause a change in the working-point as compared to the video sources.

4 CAC Based on the MSA

In this section we first compare the performance of an MSA-based CAC scheme with that of an LBA-based algorithm. Finally we attempt to overcome some drawbacks of the MSA scheme, by improving its efficiency in terms of computational time.

4.1 The CAC Model

The algorithm we implemented combines estimates of the current link utilisation with the peak rate declared by the new call to judge whether the available bandwidth is sufficient to satisfy the QoS requirements, expressed in terms of overflow probability. Figure 4 shows the scheme of the CAC algorithm. A set of calls is multiplexed at the input of a single service queue with deterministic service rate C. A measurement system estimates the bandwidth necessary to serve the active calls in order to meet their QoS requirements. This estimation is taken using both the MSA and the LBA. When a new request arrives, the new call declares a peak rate. The latter is summed to the estimated current utilised bandwidth. If the total is less than the output link capacity C, then the new call is accepted. In this way we can statistically guarantee the required QoS to all connections.

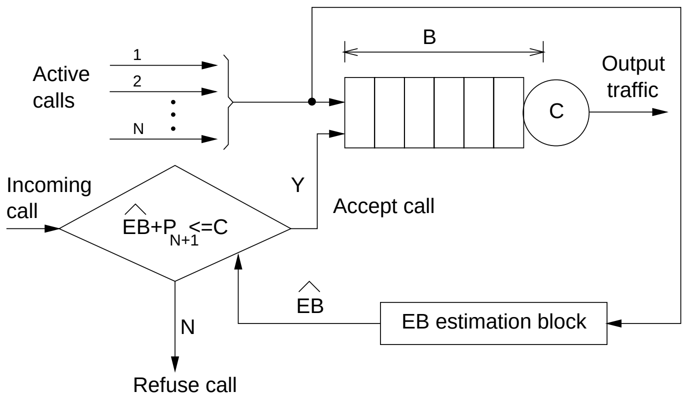


Fig. 4. AC block diagram scheme

As soon as the new call starts transmitting data, the algorithm begins to update the estimation of the total bandwidth requirement. This estimate has to be updated also every time a connection ends. If a new connection request arrives before the algorithm has achieved an accurate estimate of the bandwidth requirement, the algorithm acts conservatively. Indeed, it uses the most recent stable estimate of the bandwidth requirement, plus the sum of the peak rates of all subsequently admitted calls.

As already mentioned, the CAC algorithm can estimate the bandwidth requirement of the multiplexed sources using either the MSA or the LBA. A significant drawback of the MSA approach is that theoretically any time a connection starts or ends the system has to evaluate a new working-point. This requirement makes the MSA-based scheme time consuming as compared to the one based on the LBA. The latter method does not require to update the working-point as it depends only on the buffer size and the QoS constraint and it is independent of the traffic mix (see section 2.2). Therefore we tried to take advantage of the results in section 3 to speed-up the update process. We observed in fact that only a significant change in the traffic mix modifies the working-point. Therefore we propose to place proper thresholds to delimit the variation of mix beyond which a new operating-point estimate is necessary. In the last part of this section we report simulation results to show the performance achieved in this case.

4.2 Simulation Scenarios

In each simulation we modelled a single output buffer and transmission link of an ATM switch. The link speed used is 155Mbps and the buffer size is either 500 (small buffer) or 5000 cells (large buffer). The simulated time is 10,800 sec (corresponding to three hours) and the QoS requirement corresponds to an overflow probability less than 10^{-6}. To compare the performance of the MSA scheme with that of the LBA scheme, we studied three different scenarios:

1. high degree of multiplexing and large buffer size;
2. high degree of multiplexing and small buffer size;
3. low degree of multiplexing and large buffer size.

With high degree of multiplexing we mean that in the steady-state the number of multiplexed sources is at least 1000 while we refer to low degree of multiplexing when the number of multiplexed sources is about 300. Note that the first scenario satisfies both the MSA and the LBA hypothesis while the second and the third satisfy only the MSA and the LBA hypothesis respectively.

Our simulations involve the three different types of sources (voice, data and video traffic) described in section 3. The calls of a particular traffic type arrive according to an exponential inter-arrival time distribution and have an exponentially distributed length. Table 3 shows the frequency of connection requests and connection ends for each traffic type.

	Voice	Data	Video
Fre. of connection request (calls/sec)	5	5	0.5
Freq. of connection end(deaths/sec)	0.005	0.003	0.00027

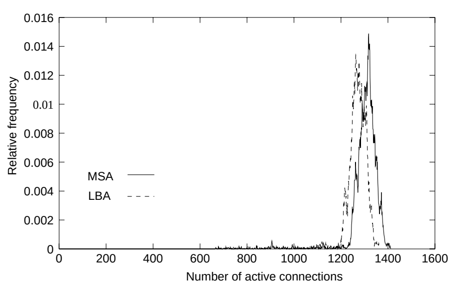
Table 2.

Each accepted call transmits a trace, derived by randomly selecting a starting point in the corresponding traffic trace. Calls are not correlated.

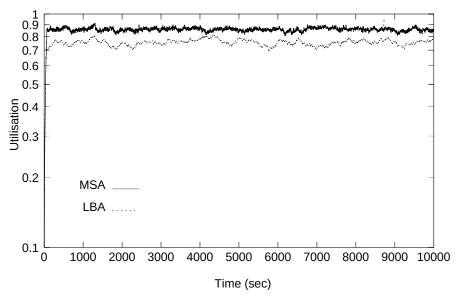
4.3 Simulation Results

In order to illustrate the results we reported for each type of simulation a graph that represents the histogram of the accepted connections and the plot of the corresponding link utilisation as a function of time. Figure 5 refers to a high degree of multiplexing and a large buffer size. Although this scenario satisfies the hypotheses of both methods, observe that the performance of the MSA algorithm clearly overcomes the LBA scheme.

Figure 5.a shows that the gap between the most probable number of connections is about 70/80 sources (7-8%) and the difference between the maximum number of accepted connections is about 50 units (5%).



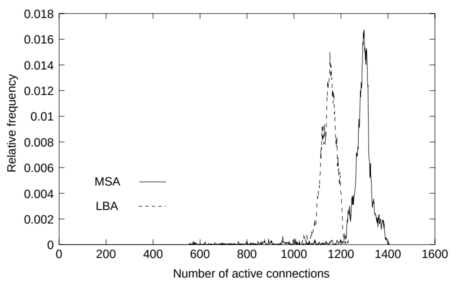
(a) Histogram of active connections



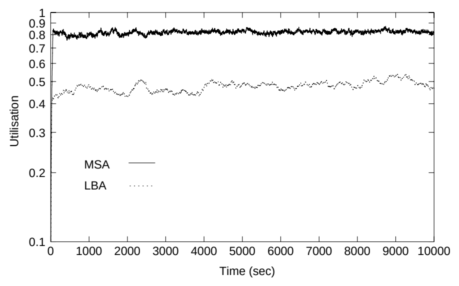
(b) Normalised accepted load

Fig. 5. Comparison between MSA and LBA based CACs : high degree of multiplexing and large buffer size

Figure 5.b shows the behaviour of the link utilisation as a function of time: note that, using the MSA method, the link utilisation approaches 1, that is the available transmission capacity is most exploited.



(a) Histogram of active connections



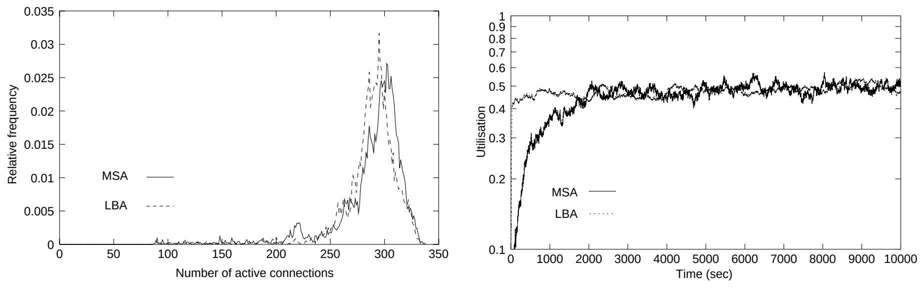
(b) Normalised accepted load

Fig. 6. Comparison between MSA and LBA based CACs : high degree of multiplexing and small buffer size

Figure 6 shows the results concerning a high degree of multiplexing and a small buffer size. As expected under these conditions, the MSA-based algorithm performs much better than the LBA-based CAC algorithm. Figure 6.a shows that the difference between the most probable number of connections and between the maximum number of accepted connections is about 200 units (20%).

Figure 6.b underlines the different utilisation reached in the steady-state: when using the LBA, the steady-state value of the link utilisation is about 0.5;

whereas by using the MSA, the steady-state value of the link utilisation amounts almost to 0.9. Note that this scenario is likely to occur in a real environment, since both the number of connected sources and the size of the buffer are realistic, especially for real time and delay sensitive applications.



(a) Histogram of active connections (b) Normalised accepted load

Fig. 7. Comparison between MSA and LBA based CACs : low degree of multiplexing and large buffer size

Figure 7 reports results for a low degree of multiplexing and a large buffer size. According to the theory, this scenario should favour the LBA-based CAC algorithm. However, figure 7 witnesses that the performance of the two methods is almost equivalent: the mean values of the histograms of the number of connected sources are very close. The same holds for the link utilisation plots.

In all three simulation scenarios the MSA-based algorithm allows either to achieve the best performance or it performs the same as the LBA scheme in terms of link exploitation. Moreover, the performance of the MSA-based algorithm is independent from the buffer size, as it results, by comparing figures 5 and 6. However, our simulations highlighted also that a major drawback of the MSA approach is the computational time necessary to evaluate the working-point, every time a connection starts or ends. Indeed, the simulations performed to test the MSA scheme resulted about sixty times longer than those for the LBA scheme.

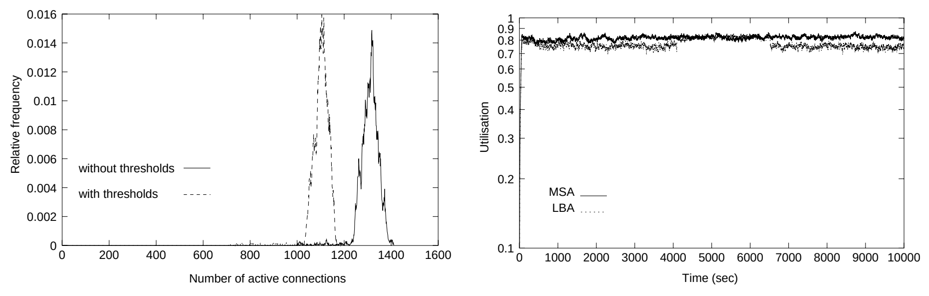
4.4 Improving the Computational Efficiency of the MSA-Based Scheme

In the following we face the problem of the computational inefficiency of the MSA-based scheme, due to the large number of updates of the working-point. Our idea is motivated by the results in section 3, that showed how only a significant change in the traffic mix modifies the working-point. Therefore, we propose to place proper thresholds to delimit the variation of mix beyond which a new operating-point estimate is necessary. Based on the results in section 3, we empirically derived thresholds as reported in table 4.

	Voice	Data	Video
Threshold	10	5	1

Table 3.

Observe that the values of the thresholds depend on the source type. We subsequently ran a set of simulations, to evaluate the degradation of the performance in terms of number of accepted connections and link utilisation. To this end, we compared results obtained by both evaluating the working-point any time a change in the mix occurs and by using thresholds as explained above. With the thresholds we selected, the simulations resulted about five times faster than those without thresholds.



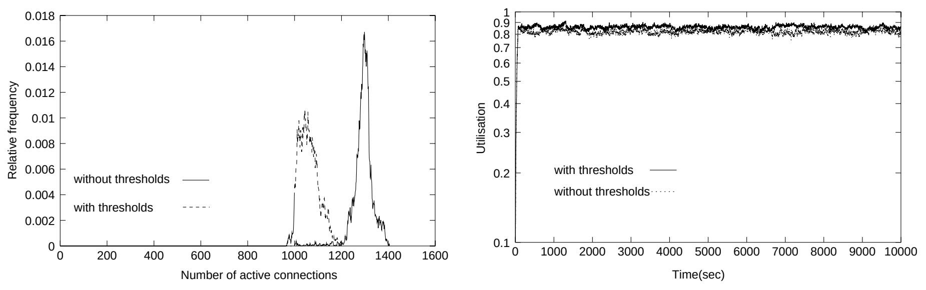
(a) Histogram of active connections (b) Normalised accepted load

Fig. 8. Comparison between MSA (with thresholds) and LBA based CACs : high degree of multiplexing and small buffer size

Figure 8 reports results derived by considering a high degree of multiplexing and a large buffer size. Note that the values for the mean and the maximum number of accepted connections clearly degenerate. However, the performance of the MSA-based CAC algorithm remains almost unchanged in terms of link utilisation when applying the proposed method.

We repeated the same test considering a high degree of multiplexing and a small buffer size. Also in this case we observed a similar behaviour (see figure 9).

This behaviour is due to a change in the mix composition of the multiplexed sources that occurs when the CAC algorithm uses thresholds. In this case, the



(a) Histogram of active connections (b) Normalised accepted load

Fig. 9. Comparison between MSA based CACs with and without thresholds

mix composition is influenced by the value of the selected thresholds: sources with a low threshold value prevail in number with respect to calls with higher thresholds. In our scenario, the selected thresholds favour the video sources. Thus, although the total number of multiplexed sources is smaller, the link utilisation achieves the same level as in the case without thresholds. This is due to the higher percentage of video sources which require a greater amount of bandwidth compared to data and voice streams.

5 Conclusions

We studied the performance of a CAC algorithm under two different mathematical approaches: the MSA and the LBA. From our analysis it turns out that the CAC scheme based on the MSA allows to achieve the best performance as compared to the LBA method. However, the MSA-based technique has the drawback of being computationally more complex, since it requires to evaluate a new operating-point every time there is a variation in the traffic mix. In order to improve the computational efficiency of the MSA-based scheme we investigated the feasibility of placing proper thresholds in the traffic mix composition. The system triggers the computation of a new working-point, only after reaching one of these thresholds. We empirically set the values of such thresholds on the basis of the analysis we carried out on the sensitivity of the working-point to traffic mix variations. By employing this technique we significantly reduced the computational effort. Besides, our simulations show that the link utilisation level remains unaffected, although with our choice of the thresholds the total number of admitted connections decreases.

References

1. D. D. Botvich and N. Duffield. Large deviations, the shape of the loss curve, and economies of scale in large multiplexers. Queuing Systems, 1995.

2. D. D. Botvich, T. J. Corcoran, N. G. Duffield, and P. Farrell. Economies of scale in long and short buffers of large multiplexers. Proc. of the 12th IEE UK Teletraffic Symposium, Paper 10 ppl-8, Old Windsor, 15-17 March 1995.
3. F. P. Kelly. Notes on effective bandwidths. In F.P. Kelly, S. Zachary, and I.B. Ziedins, editors, Stochastic networks: Theory and Applications, Royal Statistical Society Lecture Note Series 4, pages 141-168. Oxford University Press, 1996.
4. C. Courcoubetis and R. Weber. Effective bandwidths for stationary sources. Probability in the Engineering and Information Sciences, 9, 285-298,1995.
5. C. Courcoubetis, V. A. Siris, and G. D. Stamoulis. Application of the many sources asymptotic and effective bandwidths to traffic engineering. Telecommunication Systems, 1999.
6. N. G. Duffield, N. O'Connell. Large deviations and overflow probabilities for the general single-server queue, with applications. Math. Proc. Cambridge Philos. Society, Vol. 118, No. 2, pp. 363-374, 1995.
7. N. G. Duffield, J. T. Lewis, N. O'Connell, R. Russell and F. Toomey. Entropy of ATM Traffic Streams: A Tool for Estimating QoS Parameters. IEEE JSAC, Vol. 13, No. 6, pp.981-990 August 1995.
8. W. -C. Lau, A. Erramilli, J. L. Wang, W. Willinger. Self-Similar Traffic Generation: the Random Midpoint Displacement Algorithm and its Properties. Proc. of the ICC'95, Seattle, WA, pp. 446-472, 1995.
9. S. Tartarelli, M. Falkner, M. Devetsikiotis, I. Lambadaris and S. Giordano. Empirical Effective Bandwidths. Proc. of Globecom 2000, San Francisco, California, 27 November- 1 December 2000 (To appear).
10. R. J. Gibbens and F. P. Kelly. Measurement-based connection admission control. In 15th International Teletraffic Congress Proceedings, June 1997.
11. R. J. Gibbens, Y. C. Teh. Critical time and space scales for statistical multiplexing. Proc. of the 15th International Teletraffic Congress. ITC 15, Washington DC, June 1997.

Call Admission Control and Routing of QoS-Aware and Best-Effort Flows in an IP-over-ATM Networking Environment

Raffaele Bolla[1], Franco Davoli[1], Mario Marchese[2], and Marco Perrando[1]

[1] Department of Communications, Computer and Systems Science (DIST)
University of Genoa
Via Opera Pia 13, I-16145 Genova, Italy
`(lelus, franco, perr)@dist.unige.it`

[2] Italian Consortium for Telecommunications (CNIT)
University of Genoa Research Unit
Via Opera Pia 13, I-16145 Genova, Italy
`mario.marchese@cnit.it`

Abstract. In the context of an IP-over-ATM access and transport network, carrying guaranteed quality (CBR, rt-VBR) services as well as IP datagrams (as ABR or UBR traffic classes), we consider the joint problems of Call Admission Control (CAC), bandwidth allocation and routing. The presence of distributed access multiplexers is assumed, which are both geographically dispersed (e.g., at the user premises) and hierarchically structured. Such multiplexers are intelligent devices with decision making capabilities that operate jointly, in order to make the best possible use of the transport capacity of the access network and to maintain the Quality of Service (QoS) requirements of different users and service classes. Following the physical system organization, a hierarchical control structure is defined, where the admission of calls for real-time traffic classes (or different users) is performed by independent controllers; the latter are parametrized by the bandwidths allocated by a common link agent, playing the role of a "link coordinator" in the hierarchical control scheme. This decision maker aims at minimizing a general cost that captures QoS requirements both at the call-level (call blocking probability) for QoS-aware, connection-oriented services and at the cell-level (cell loss probability) for connectionless, best-effort, ones. The control architecture also reflects the multilayer hierarchy introduced by the presence of multiple teletraffic time scales, by essentially decoupling the above problem from that of ensuring QoS at the cell-level for services of the first type. We derive the optimal parameters' setting from the numerical solution of a mathematical programming problem. Then, the same structure is applied to link multiplexers of the transport network nodes, which are supposed to possess both ATM and IP switching/routing capabilities. Routing strategies at both ATM and IP levels are defined, which are combined with the above described CAC and bandwidth allocation scheme. The performance of the whole structure is tested by simulation.

1 Introduction

Key factors for multimedia telecommunication services distribution to business and residential customers on a large scale are the presence of a broadband access network [1,2,3,4,5] and of an integrated high-speed transport.

In general, a similar framework can be recognized in several access networks, where distributed multiplexing units are present, which are both geographically dispersed (e.g., at the user premises) and hierarchically structured. With current computing technology, such multiplexers can become intelligent devices with decision making capabilities, which operate jointly in order to make the best possible use of the transport capacity of the access network, while at the same time maintaining the Quality of Service (QoS) requirements of different users and service classes. In this respect, a first distinction can be made between QoS-aware, connection-oriented services, and connectionless, best-effort, ones. At the same time, whenever cell-level statistical multiplexing is present for services of the first type, the control architecture should also reflect the multilayer hierarchy introduced by the presence of multiple teletraffic time scales, by coping with the problem of ensuring QoS at the cell-level also in this case (cell loss probability and, possibly, cell transfer delay). A similar structure for service and time-scale decoupling can be recognized also at link multiplexers within a switching node in the transport network, be it based on IP only or on IP-over-ATM.

These features suggest the application of dynamic hierarchical control structures, similarly as in the context of hybrid TDM [6], where the admission of calls for real-time traffic classes (or different users) is performed by independent controllers; the latter are parameterized by the bandwidths allocated by a common link agent, playing the role of a coordinator in the hierarchical control scheme. This agent's decisions are based on the (constrained) minimization of a global cost, whose form tries to capture QoS requirements, both at the call-level (call blocking probability) for QoS-aware, connection-oriented services, and at the cell-level (cell loss probability) for connectionless, best-effort, ones.

In this paper, we consider a general structure of such multiplexers, loaded with different service classes, with the goal of defining management and control laws and algorithms of the type mentioned above in this specific context. To this aim, we suppose to operate over an ATM access and transport structure, and to have a mix of Continuous Bit Rate (CBR) and Variable Bit Rate (VBR) service classes, which share a common bandwidth with Available Bit Rate (ABR) or even Unspecified Bit Rate (UBR) ones; the latter are generated by a flow of data packets at the network layer (e.g., IP datagrams with best-effort service). Given the bandwidth allocated to the CBR and VBR sources, we derive the region in the space of connections of the various classes of this type, within which cell-level QoS is satisfied (Service Separation with Dynamic Partitions [7]): this concept essentially decouples the problem of cell- and call-level QoS. Above this region, the dynamics of connection-oriented, QoS-aware, traffic will be described by Markov chains at the call level, and Call Admission Control (CAC) strategies will be defined. On the other hand, a self-similar traffic model can be used to characterize connectionless, best-effort, traffic (e.g., "short-lived" IP flows), which may be given the "residual" bandwidth over the link, possibly with a constraint on the minimal allocation (in order to avoid TCP congestion control to drastically reduce the throughput). We use the above models to also construct a coordination structure, which is based on a hierarchical decomposition between "local" admission controllers and a link bandwidth allocation controller that plays the role of the coordinator. In the access area, this hierarchical structure may be applied to realize a decomposition among both service classes and users. At the transport nodes, only decomposition and coordination among the services for each outgoing link is present.

After the introduction and analysis of this CAC and bandwidth allocation paradigm, we also define routing strategies, both for connection-oriented, QoS-aware traffic at the ATM call-level (including "long-lived" IP flows) and for connectionless "short-lived" IP flows. To do so, we suppose switching nodes to possess both IP and ATM switching capabilities; ATM VP/VC routing is used for traffic of the first type, whereas datagrams of the second type are segmented into cells and transferred over ATM VPs between IP routers, where the datagram is reconstructed and routed accordingly. In particular, in the present work we suppose to use the underlying ATM structure to ensure QoS, by only considering best-effort IP services with a minimum assured quality. However, the concepts used can be applied also in different scenarios, e.g., by considering IP flows with different QoS (as in the DiffServ paradigm) over a MPLS platform, when bandwidth allocation is performed. The definition of a similar architecture in this environment is currently under investigation; in fact, admission and congestion control issues in the Internet environment are among the most challenging topics in the recent literature (see, e.g., [8,9,10]).

The paper is organized as follows. We outline our model and the cell-level requirements in the next section. Section 3 is dedicated to the admission control level, and to the definition of the cost function of the bandwidth allocation level. Section 4 extends the model to the case of a complete access and transport network, by introducing also the proposed solution for routing. Numerical simulation results on the performance of the overall control scheme are reported and discussed in Section 5. Section 6 contains the conclusion.

2 Traffic Models and Service Separation

We suppose the traffic in the network to be categorized into $H+1$ service classes. The first H contain either CBR or bursty VBR (on-off) sources, characterized by statistical parameters like peak rate, average transmission rate and average burst length, as well as by QoS requirements, like cell loss probability and cell delay. We indicate with $B^{(h)}$ [cells], $P^{(h)}$ [bits/s], $M^{(h)}$ [bits/s] and $b^{(h)} = P^{(h)}/M^{(h)}$, the average burst length, the peak bit rate, the average bit rate and the burstiness, respectively, of a source of the h-th class, $h = 1, \dots, H$ (obviously, CBR sources are included in this description, with $b^{(h)} = 1$). We let $\lambda^{(h)}$ and $1/\mu^{(h)}$ represent the average arrival rate and the average duration of connections of class h, respectively, and $\rho^{(h)} = \lambda^{(h)}/\mu^{(h)}$. The channel time is slotted, and a slot carries an ATM cell.

Moreover, we suppose to have an asynchronous packet flow, which represents the traffic generated by *connectionless*, best-effort, services; this flow is supposed to originate from the superposition of a number of on-off sources, whose sojourn time Y (expressed in "source time units", to be defined below) in the active state follows a Pareto distribution, i.e.,

$$\mathbf{Pr}\{Y = y\} = cy^{-(\alpha+1)} \quad 1 < \alpha < 2, y \geq 1 \tag{1}$$

where c is the normalization constant and α is a parameter. In our setting, the "source time unit" T is defined as the packetization delay of the above-defined sources, i.e., the time to generate a cell at the source speed.

The Pareto distribution is well known for its "heavy-tail" property, and has actually been used to model self-similar traffic; more specifically, the aggregation of a large number of sources of the above mentioned type has been shown to give rise to self-similar traffic [11]. The packets (after segmentation into ATM cells) receive a variable rate service (ABR or even UBR). We model the queueing of cells generated in this way as a synchronous $Z/D/C_{asy}/Q^{(H+1)}$ system, where Z is the aggregate self-similar process (discretized over a "source time unit"), C_{asy} the capacity available for the connectionless traffic, and $Q^{(H+1)}$ [cells] is the dimension of the buffer dedicated to it. In the following, the upper bound on the overflow probability derived in [11] will be used.

At each ATM multiplexer, traffic class h, $h = 1, \ldots, H$, is assigned a separate buffer of length $Q^{(h)}$ [cells], whose output is statistically multiplexed on the outgoing link by a scheduler, which substantially divides the part of channel capacity assigned to connection-oriented traffic C_{co} [bits/s] into "virtual" partitions $C^{(h)}$ among the classes, whose sum amounts to C_{co} (service separation). The partitions may be maintained by serving the buffer in a Weighted Round Robin fashion, or by using a technique like Generalized Processor Sharing [12]. The overall queueing system at the cell level is depicted in Fig 1.

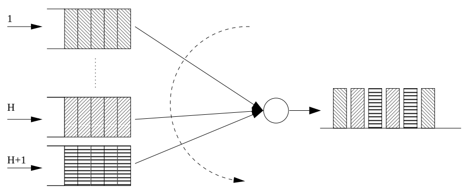


Fig. 1. Cell-level multiplexing under service separation.

Connection requests are also processed on a per-class basis. Given a model for the traffic sources of a class, the cell-level performance requirements (e.g., in terms of average cell loss and delayed cell rate) allow to define a region in call-space (which will be referred to as "Feasibility Region" or FR), where they are certainly satisfied. This region corresponds to the CAC method named "service separation with dynamic partitions" in [7, p.147]. In a network, one such region can be associated with each link.

Clearly, the points on the boundary of the FR correspond to the maximum numbers of Virtual Circuit (VC) connections $[N_{max}^{(1)}, \ldots, N_{max}^{(H)}]$ that are compatible with the given cell-level QoS constraints. We can associate each $N_{max}^{(h)}$ with the minimum amount of bandwidth $C_{min}^{(h)}$ that is necessary to support that number of connections with the given QoS guarantees.

The computation of the FR has been the object of several studies and can be effected in different ways, either by analysis, given a model of the traffic sources, or by simulation. Using any approach based on equivalent bandwidth (involving, in our Service Separation context, only homogeneous sources) yields a straightforward boundary of the FR. In any case, it is worth noting that, in the context of the methods to be considered in the next section, the FR itself will be just a tool to describe the CAC schemes. The specific technique to ensure QoS satisfaction at the cell-level might be changed (always within the framework of Service Separation), without affecting the access control general procedure.

However, to fix ideas, we refer here for the computation of the FR to the model we have used in [13] and in previous works, where a maximum threshold value $\epsilon^{(h)}$ is set for the average cell loss rate $(P_{loss}^{(h)}(N^{(h)}, C^{(h)}))$ and another one $(\delta^{(h)})$ for the average delayed cell rate $(P_{delay}^{(h)}(N^{(h)}, C^{(h)}))$, with $N^{(h)}$ accepted calls and a bandwidth $C^{(h)}$ assigned to traffic class h. An Interrupted Bernoulli Process (IBP) [14] is used to model the state of a call, from which $P_{loss}^{(h)}(N^{(h)}, C^{(h)})$ and $P_{delay}^{(h)}(N^{(h)}, C^{(h)})$ are derived, under a quasi-stationarity assumption on the number of active connections, and both $N_{max}^{(h)}$ and $C_{min}^{(h)}$, $h = 1, \ldots, H$, are computed. We remark again that, as an alternative approach, any one based on equivalent bandwidth (e.g., [15]) could be used, yielding similar results (see [13]). Obviously, in the present case, the FR is parametrized by the capacity C_{co} actually assigned to the connection-oriented traffic.

We let

$$\mathbf{N}_A(k) = \mathbf{col}[N_A^{(h)}(k), h = 1, \ldots, H] \tag{2}$$

where $N_A^{(h)}(k)$ is the number of connections in progress at the generic instant (slot) k for class h. The vector in (2) represents the state of the system at instant k (the VC-profile in [7]).

3 Call Admission Control and Capacity Allocation

At this point, we are ready to consider QoS performance measures and constraints at the call level, having essentially decoupled this problem from the lower level one. As in our previous work in the ATM context [13], we have chosen to adopt admission control policies for our connection-oriented traffic that belong to the class of Complete Partitioning (CP) ones; such policies define a "rectangular" sub-region within the FR, by dividing the available capacity among traffic classes in a "static" way. In other words, given a point $\mathbf{N}^*_{max} = \mathbf{col}[N^{*,(h)}_{max}, h = 1, \ldots, H]$ on the boundary of a specific FR (as defined in the previous section), a new connection of class h at time k is accepted only if

$$N^{(h)}_A(k) + 1 \leq N^{*,(h)}_{max} \tag{3}$$

Even with this restriction, there are several optimization criteria that can be followed, in order to place the vertex $\mathbf{N}^*_{max}$ of the rectangular acceptance region [13]; in the present case, an additional constraint may be added by the requirements of the connectionless traffic. Actually, even though this traffic will be mostly best-effort, an upper bound on the cell loss probability may be considered in the bandwidth allocation, in order to avoid some undesired effects (e.g., too many TCP retransmissions).

Before considering the choice of the "optimal" point $\mathbf{N}^{opt}_{max}$, we may note that the connectionless traffic can always be allocated all the bandwidth unused by the connection-oriented classes, in a way analogous to movable boundary schemes in TDM networks [6]. We can do this through the knowledge of the VC-profile $N^{(h)}_A(k)$, by calculating (with any valid method, as mentioned in the previous section) the minimum bandwidth $C^{(h)}_{min}(k)$ that is necessary to ensure cell-level QoS to the $N^{(h)}_A(k)$ connections of class h in progress. By letting

$$C_{min}(k) = \sum_{h=1}^{H} C^{(h)}_{min}(k) \tag{4}$$

we can assign (through the scheduler) the connectionless traffic the *residual bandwidth*

$$R(k) = C - C_{min}(k) \tag{5}$$

where C is the total transfer capacity of the link; this assignment can last until either a new call is accepted or a connection terminates. It can be noted that, given the connection-oriented traffic characteristics and the bandwidth C_{co} globally assigned to it, the corresponding FR can be constructed off-line, and the residual bandwidth $R(k)$ can be determined for each of its points.

Even with the above described assignment, it is however clear that different "static" partitions (determined by the value of C_{co}) may be necessary to combine the requirements of the various classes. These will be determined by the optimization procedure that leads to the choice of $\mathbf{N}^{opt}_{max}$. By first setting $C_{co} = C$ and then gradually decreasing (in discrete steps) the bandwidth globally allocated to connection-oriented services, we obtain a family of FRs, with decreasing "volumes". For each corresponding boundary $S(C_{co})$, we can compute the point $\mathbf{N}^*_{max}(C_{co}) \in S(C_{co})$ that minimizes a cost function involving the stationary call blocking probabilities $P^{(h)}_{block}(N^{(h)}_{max})$; owing to the service separation assumption, each of these probabilities can be simply expressed by the Erlang B formula [7], where $N^{(h)}_{max}$ is the number of servers. Among the various possibilities, we have chosen the following cost function (named Balanced Erlang Scheme (BES) in [13], which tends to equalize weighted blocking among the classes):

$$J_1(\mathbf{N}_{max}) = \max_h \{\theta^{(h)} P^{(h)}_{block}(N^{(h)}_{max})\} \tag{6}$$

If, for each service class, we also want to take into account a constraint on the blocking probability, say

$$P^{(h)}_{block}(N^{(h)}) \leq \mathit{\Gamma}^{(h)} \quad h = 1, \dots, H \tag{7}$$

we can modify the cost function (6) as follows. Let $\bar{\mathbf{N}}$ be the point whose coordinates satisfy relations (7) with equality; then we can consider

$$\tilde{J}_1(\mathbf{N}_{max}) = \begin{cases} \tau J_1(\mathbf{N}_{max}) & \mathbf{N}_{max} < \bar{\mathbf{N}} \\ J_1(\mathbf{N}_{max}) & \mathbf{N}_{max} \geq \bar{\mathbf{N}} \end{cases} \tag{8}$$

where τ is a constant (the larger τ, the higher the penalty for not matching the constraint), and the inequalities involving vectors are to be interpreted as applied to all the components.

We now turn to the definition of a cost function related to the performance of the best-effort traffic.

Let R represent a value of capacity (expressed in slots per "source time unit") available for the asynchronous traffic. Moreover, let $a_Y \equiv \mathbf{E}\{Y\} = c\sum_1^\infty y^{-a}$ be the mean of the Pareto distribution (1), P_{over} be the probability of buffer overflow, and let λ indicate the intensity of the number of sources becoming active at a generic instant ("source time unit") in the aggregated process [11].

If $\lambda a_Y < R$, then

$$P_{over} \leq \frac{c\lambda}{\alpha(\alpha-1)(R - a_Y\lambda)} (Q^{(H+1)})^{-\alpha+1} \tag{9}$$

asymptotically with $Q^{(H+1)}$ [11].

We can define

$$P_{loss}(R) = \begin{cases} \min\{\frac{c\lambda}{\alpha(\alpha-1)(R-a_Y\lambda)}(Q^{(H+1)})^{-\alpha+1}, 1\} & \text{if } R > a_Y\lambda \\ 1 & \text{otherwise} \end{cases} \tag{10}$$

as an approximation of the loss probability.

Given $\mathbf{N}^*_{max}(C_{co}) = \arg\min \tilde{J}_1[\mathbf{N}_{max}(C_{co})]$, for each value of C_{co}, the average loss probability for the asynchronous, connectionless traffic can be computed as:

$$\bar{P}^{(H+1)}_{loss}(\mathbf{N}^*_{max}) = \sum_{\mathbf{s} \in S_R(\mathbf{N}^*_{max})} P_{loss}(R_\mathbf{s})\pi(\mathbf{s}) \tag{11}$$

where $R_\mathbf{s}$ represents the residual capacity corresponding to the value $\mathbf{s}$ of the VC-profile $\mathbf{N}_A(k)$ (the system's state) and $S_R(\mathbf{N}^*_{max})$ is the "rectangular" region within the FR, whose vertex on the FR's boundary is $\mathbf{N}^*_{max}$. With the given CAC rule, the state of the system $\mathbf{N}_A(k)$ is a multidimensional Markov chain over $S_R(\mathbf{N}^*_{max})$, whose stationary distribution has been indicated by $\{\pi(\mathbf{s})\}$.

Essentially, in determining both P_{loss} and the stationary distribution $\{\pi(\mathbf{s})\}$, one can use the same decoupling procedure as in [6] and [16]; $\{\pi(\mathbf{s})\}$ is computed from the transition probability of the multidimensional birth-death process describing the call dynamics [6].

Equation (11) may represent a possible choice for the asynchronous traffic cost function (and has been indeed adopted in [17,18]). A simpler alternative may be constituted by the maximum loss probability incurred by the best-effort traffic with a given choice of C_{co} (and consequently of $\mathbf{N}^*_{max}$), namely

$$P^{(H+1)}_{loss,max}(C_{co}) = P_{loss}(R_{co}) \tag{12}$$

where R_{co} is the number of slots per source time unit corresponding to the bandwidth $C - C_{co}$.

In order to simplify the computational effort, in this paper we have chosen the cost function for the best-effort traffic as

$$J_2[C_{co}] = P^{(H+1)}_{loss,max}(C_{co}) \tag{13}$$

Here, we may want to set a desired level of (rather than a strict constraint on) the maximum cell loss that is tolerable for the connectionless service, i.e.,

$$P^{(H+1)}_{loss,max}(C_{co}) \leq \epsilon^{(H+1)} \tag{14}$$

We can now turn to the determination of the "globally" optimal point N^{opt}_{max} that minimizes an overall cost function, to be defined. Among a few possible choices (see [18] for a different one), we have found that the following allows to obtain the most straightforward control over the desired tradeoff between the two traffic types; specifically, we have chosen

$$\mathbf{N}^{opt}_{max} = \arg\min_{C_{co}}\{\max\{\tilde{J}_1[N^*_{max}(C_{co})], \sigma J_2[C_{co}]\} \tag{15}$$

where σ is a weighting coefficient. The above form of the overall link cost tends to equalize the two values at the minimum point; in any case, their relative importance is weighted by the parameter σ, whose choice can take into account the "desired" upper bound in (14).

All calculations related to the FRs only depend on the cell-level parameters, i.e., on the statistical characteristics of the sources, which determine their categorization into specific classes, but not on the call-level ones (e.g., offered load); for a given classification, all the FRs can be computed off-line. The same would be possible for the optimization we have described, if the offered load of each traffic class remained constant: actually, this is the situation that was considered in the numerical examples reported in [17], [18], where some instances of the feasibility regions and of the corresponding optimal points have been derived. However, slow variations in the offered load of some class could be followed by an on-line application of the algorithm, possibly by adopting some simplified numerical procedure, along the same lines as in [6]. Such on-line bandwidth reallocation will be applied in conjunction with the routing strategy, in the network-wide control architecture described in the following section.

It is also worth noting that the case considered in this paper is limited to the inter-class bandwidth allocation, where a service class gathers the flows generated by multiple users; a similar optimization procedure can be extended to include explicitly the service provision to individual users (or user aggregations) at the access nodes, and can distinguish explicitly the downlink and uplink directions. This requires a further (third) level of coordination, which allocates the corresponding bandwidth shares, according to a new overall link cost function. This problem has been touched in [19], with respect to an ATM-Passive Optical Network (ATM-PON) model [5] and is also the subject of further investigation.

4 Combining CAC, Bandwidth Allocation, and Routing

As we mentioned, the CAC and bandwidth allocation technique previously introduced can be applied not only at the network access multiplexers, but also within each node. More specifically, we suppose to have a controller of this kind for each output link of the switching node. However, we need now to combine their actions with those of the routing strategies. We have taken into account the Distributed Least Congested Path (DLCP) algorithm [19], [20] for routing of incoming calls at set-up time, and a minimum hop routing for the asynchronous traffic. The two algorithms work jointly with distributed bandwidth allocators, one for each link at each network node, operating as adaptive controllers, whose aim is to adjust the bandwidth sharing among traffic classes, to absorb possible variations in the traffic flows caused by the routing decisions.

The DLCP routing works as follows. At connection set-up, a call request packet is forwarded, hop by hop, from node to node. At each node traversed, the call request of a certain class is first assigned a subset of outgoing links towards the destination, upon which the resources necessary to maintain the required level of QoS are available; then, a routing decision is taken among these links, by choosing the least “cost” one.

The cost of the link is formed by two different terms: the first one states the actual (local) link congestion; the second one takes into account, in the same way, an aggregate (global) congestion measure of the successors of the link. A node i chooses the link to which to forward a class h connection request, by minimizing (over the subset of outgoing links ij that lead toward the destination required) the quantity

$$c_{ij}^{(h)}(k,s) = c_{ij,L}^{(h)}(k) + \xi_j c_j^{(h)}(s) \tag{16}$$

where $c_{ij,L}^{(h)}(k)$ is the actual link congestion and $c_j^{(h)}(s)$ is the aggregate measure of node j and its successors, measured at time $s < k$ and communicated to the node; $\xi_j \in [0,1]$ is a weighting coefficient, used to balance the influence of the local and aggregate cost. For the local congestion of link ij, we choose the following form:

$$c_{ij,L}^{(h)}(k) = \frac{1}{N_{ij,max}^{(h)}(k) - N_{ij,A}^{(h)}(k) + 1} \tag{17}$$

where $N_{ij,max}^{(h)}(k)$ and $N_{ij,A}^{(h)}(k)$ correspond to the same quantities defined in sections 2 and 3 above, indexed by the link indices, having dropped the $*$ symbol where present.

The aggregate cost of node j is formed by two terms

$$c_j^{(h)}(s) = c_{j,L}^{(h)}(s) + \zeta_j c_{j,A}^{(h)}(s) \tag{18}$$

where ζ_j is a weighting coefficient, $c_{j,L}^{(h)}(s)$ represents the average situation of the node with respect to the congestion state of its links, and $c_{j,A}^{(h)}(s)$ is an aggregate information on the average congestion of its adjacent nodes. More specifically, we define

$$c_{j,L}^{(h)}(s) = \frac{1}{L_j} \sum_{k \in Succ(j)} c_{jk,L}^{(h)}(s) \tag{19}$$

$$c_{j,A}^{(h)}(s) = \frac{1}{L_j} \sum_{k \in Succ(j)} c_k^{(h)}(s) \tag{20}$$

$Succ(j)$ being the set of nodes that are successors of node j, and L_j its cardinality. The aggregate costs are exchanged periodically among adjacent nodes.

The strategy applied to route the data flows should try to avoid oscillatory phenomena due to the dynamic interaction of routing and reallocation algorithms. The most straightforward way to achieve this is to use a shortest path algorithm, with a metric that is not time varying (e.g., based on the minimum number of hops), and let the reallocation procedure ensure enough bandwidth over the links by driving connection-oriented traffic in other directions, if necessary. We have actually used this approach in deriving the simulation results below. Obviously, other routing metrics for the data flows may be adopted. Among these, one might be constituted by the mean residual capacity of each link left by the calls; this is indeed dependent on the (controlled) behavior of the QoS-aware traffic classes, but should be "smoother" than the instantaneous residual capacity (even though it requires a new route computation each time the mean residual bandwidth changes, which occurs either when the traffic intensity of the calls varies, or the bandwidth allocation changes the value $N_{ij,max}^{(h)}(k)$). Still another possibility is to adopt a metric based on the minimum capacity reserved for best-effort traffic ($C - C_{co}$). In this case, the metric remains anyway constant between bandwidth reallocation instants and its computation is simpler.

In all cases, the routing technique works together with the CAC and bandwidth allocation, through of the term $N_{ij,max}^{(h)}(k)$ that is chosen by the latter according to the previously stated scheme: $N_{ij,max}^{(h)}(k) = N_{ij,max}^{(h),opt}$, adopting, for example, the constrained BES algorithm. It must be noted that now the traffic intensities (of both calls and data), viewed at the input of the links used by the algorithm, are strictly dependent from the routing decisions. In practice, this means that they must be estimated over a time window; the corresponding partitions must be recomputed either when the estimates fall outside a given range around the value that was previously used for the bandwidth allocation, or periodically at fixed time intervals.

5 Numerical Results

Stability, fairness and efficiency of the above described reallocation and routing algorithms have been investigated by simulation, under a range of different load conditions.

A list of default test parameters follows; unless otherwise stated, the values indicated are adopted in the simulation runs.

– Connection-oriented traffic characteristics (2 classes have been used) are indicated in Table 1.

	Class 1	Class 2
Peak rate [Mbits/s]	0.0640	10
Average rate [Mbits/s]	0.0256	2
Maximum delay [μs]	1000	1000
Buffer length [cells]	100	100
Maximum cell loss rate	10e-6	10e-4
Maximum delayed cell rate	10e-5	10e-4
Mean call duration [s]	180	480
Call arrival rate [s^{-1}]	30	0.077

Table 1. QoS-aware traffic classes characteristics.

– The best-effort traffic class is characterized as an aggregate flow having an average rate of 150 Mbits/s, and a Pareto coefficient $\alpha = 1.4$.
– The network has the topology depicted in Fig. 2, where the transport capacity of each link is 150 Mbits/s.
– The successors' cost weighting parameter over the local link cost is taken as $\xi_j = 0.5$ for every link j.
– The successors' cost weighting parameter over the downstream aggregate cost is also taken as $\zeta_j = 0.5$ for every link j.
– The reallocation frequency, if periodic reallocation is assumed, is one reallocation every 100 s.

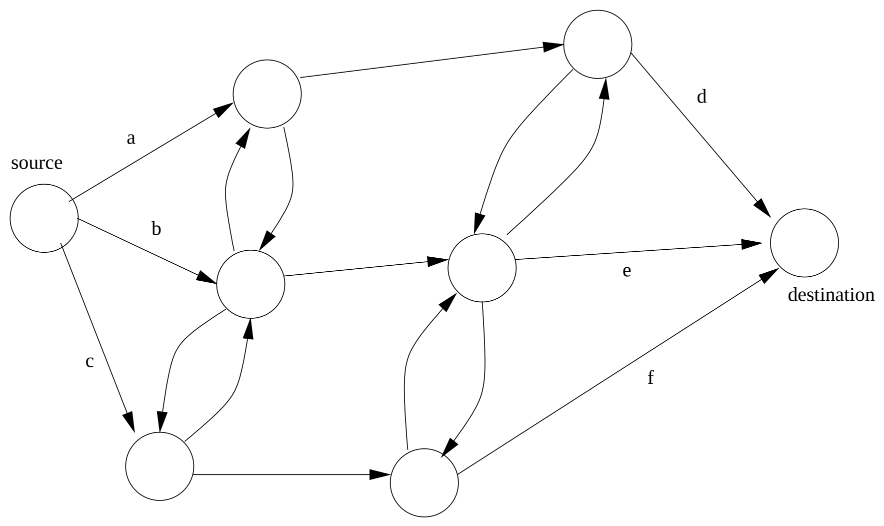


Fig. 2. The test network topology.

- The window size used to estimate the load over every link is set at 200 s.
- The network load L represents a scaling factor applied both to the call arrival rate stated before, and to the average rate of the data flow. With $L = 1$ the load in Erlangs generated by connection-oriented traffic is 5400 Erlangs for class 1 and 37 Erlangs for class 2.
- The constraints (7) have not been enforced, i.e. $\tilde{J}_1 = J_1$.
- The aggregate cost information exchange interval is 1 s.
- All the weighting coefficients $\theta^{(h)}$ in (6) have been taken equal to 1.

For what concerns DLCP routing, no preassigned path has been planned, but an algorithm avoids links that would cause loops or that would increase the number of hops towards the destination with respect to the minimum number required from the node where the decision is taken.

The data flows have not been modeled with a “single-cell” granularity (i.e., at the cell level), but a mapping has been estimated between the average load served by a given link capacity (namely, the residual link capacity) and the resulting loss rate of the data cells, in order to render simulations feasible, by generating events only at the call time scale. Rather than applying again the approximation given by (9), (10) to evaluate the loss rate, we have preferred to run a number of independent simulations “off-line”, at different values of normalized offered load, and to use their results in the definition of the mapping. During the simulation runs at the call-level in the network, the response to load values not in the mapping table is obtained by interpolation. This choice has been preferred, because it results in a smaller error than that given by the upper bound. Actually, the use of the latter can be justified in the definition of the cost function for the allocation, where an analytical expression is more handy, and a characterization of the loss behavior can be sufficient for control purpose; on the other hand, the loss rate is necessary at this stage in order to derive the real data flows at a nodes' output, which we would like to be known with as much accuracy as possible (in principle, with the same accuracy that would be given by direct measurement, i.e., by running the whole network simulation at the cell scale).

Two kinds of results have been investigated, relating to the dynamic and "average" behavior of the controlled system. The first one shows the behavior of the joint routing and allocation strategies, by displaying the allocation results for the two QoS-aware traffic classes on the input and output links of the network during the simulation time. The second one aims at showing the fair and efficient operation of the strategy, by displaying the average values of the blocking rate of the calls and of the loss rate of the data flow, for various network load values. Each point is obtained by averaging over the events occurred in a time window of 7000 s, having dropped the first 3000 s of simulation, in order to eliminate the effects of initial transients.

Fig. 3 reports the results of the bandwidth allocation over time, for the two connection-oriented traffic classes over the three input links of the test network, for a load $L = 0.7$ and a weighting coefficient $\sigma = 5000$. For the same parameter

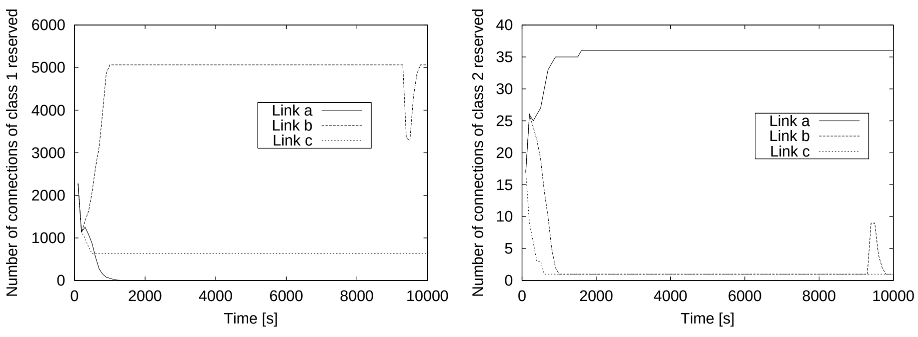


Fig. 3. Allocation for traffic class 1 and 2 over the three input links.

values, Fig. 4 shows the allocation over the three output links (d, e, f). It is worth noting that the latter is directly influenced by all the routing decisions (and indirectly by the bandwidth allocations over the previous links). The behavior is relatively steady. In particular, oscillations that may occur, due to statistical fluctuations, are readily compensated. The behavior of the allocation depends on the joint effect of bandwidth allocation, CAC, and routing. In the particular case under consideration the two connection oriented classes are almost uniquely assigned to different "initial" and "final" links, rather than being distributed among them. The link that is almost unused by these traffic types, is mostly dedicated to the best-effort one. In any case, the goal of the joint control effort is to achieve the desired balance between various performance indices (call blocking for the two connection-oriented classes and data loss probability for the best-effort one).

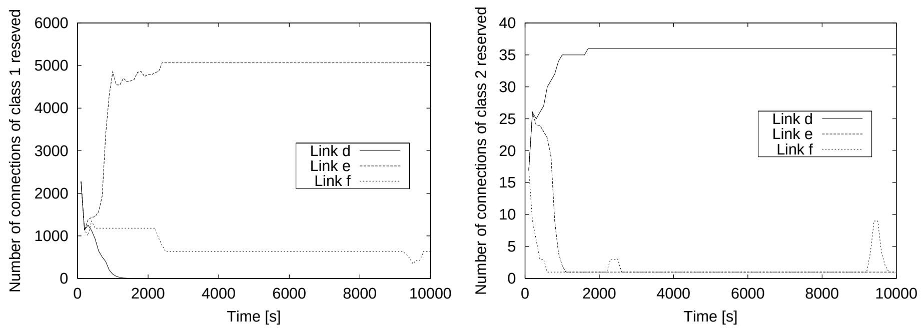


Fig. 4. Allocation for traffic class 1 and 2 over the three output links.

Fig. 5 shows the overall average blocking probability and the loss probability seen by the destination, estimated over 10000 s simulation runs (by dropping the first 3000 s). The 95% confidence intervals are also displayed. The two plots refer to the situation of synchronous and asynchronous reallocation, respectively. As expected, the latter yields slightly larger call blocking and data loss values, but still very low for loads below saturation.

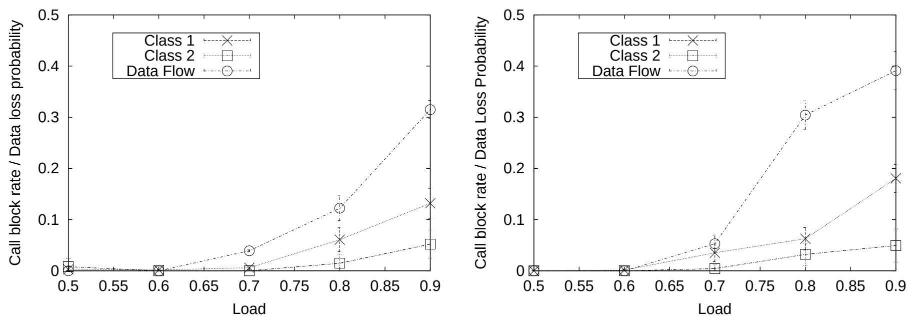


Fig. 5. Average block and loss rate with synchronous and asynchronous reallocation.

The effect of different values of the weighting coefficient σ, used in equation (15), is shown in fig. 6. Values of σ in the order of 10000 practically equalize the blocking and loss probabilities.

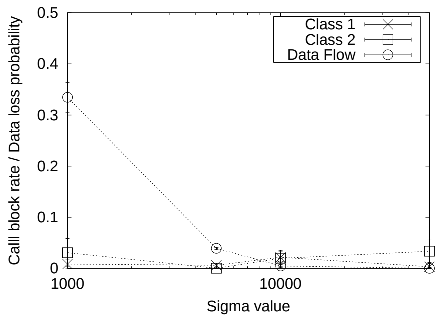


Fig. 6. Blocking and loss probability vs. coefficient σ.

6 Conclusions

We have introduced and analyzed a control scheme operating with ATM and IP over ATM service classes. The presence of multiple classes with possibly conflicting requirements has been taken into account and a joint strategy for resource (bandwidth) allocation and call admission control has been defined for each link in the network. The hierarchical control approach we have followed decouples cell- and call-level requirements and makes use of service separation with optimized and adaptive bandwidth partitioning. In particular, the presence of connectionless, best-effort, traffic with self-similar properties has been explicitly incorporated in the model. The operation of the resulting control architecture has been integrated with that of a distributed routing algorithm. The latter is actually composed of two related modules, dedicated to QoS-aware and best-effort flows, respectively. The global behavior of the whole integrated control system has been tested by simulation. In general, the resource allocation results in an adaptive flexible strategy, with a reasonable degree of stability. Moreover, the reallocation procedure can be implemented asynchronously by the links, with a very limited decrease in performance, with respect to periodic synchronous updates. Further work is being done to investigate the effect of cooperative reallocation strategies, where the link controllers share some information on their respective states.

Acknowledgment

This work was supported by the Italian Ministry of the University and Scientific and Technological Research (MURST), under the national Research Program on Techniques for QoS Guarantees in Multiservice Telecommunication Networks.

References

1. C.-J.L. van Driel, P.A.M. van Grinsven, V. Pronk, W.A.M. Snijders, “The (R)evolution of access networks for the Information Superhighway”, *IEEE Commun. Mag.*, vol. 35, no. 6, pp. 104-112, June 1997.
2. V.K. Bhagavath, “Open technical issues in provisioning high-speed interactive data services over residential access networks”, *IEEE Network Mag.*, vol. 11, no. 1, pp. 10-12, Jan./Feb. 1997.
3. D. Faulkner, R. Mistry, T. Rowbotham, K. Okada, W. Warzanskyj, A. Zylbersztejn, Y. Picaud, “The Full Services Access Networks initiative”, *IEEE Commun. Mag.*, vol. 35, no. 4, pp. 58-68, April 1997.
4. I. Van de Voorde, G. Van der Plas, “Full Service Optical Access Networks: ATM transport on passive optical networks”, *IEEE Commun. Mag.*, vol. 35, no. 4, pp. 70-75, April 1997.
5. Full Services Access Network Requirements Specification, J.A. Quayle, Ed., Issue 1, 1998, (http://www.labs.bt.com/profsoc/access/index.htm).
6. R. Bolla, F. Davoli, “Control of multirate synchronous streams in hybrid TDM access networks”, *IEEE/ACM Trans. on Networking*, vol. 5, no. 2, pp. 291-304, April 1997.
7. K.W. Ross, *Multiservice Loss Models for Broadband Telecommunication Networks*, Springer Verlag, London, UK, 1995.
8. R.J. Gibbens, F.P. Kelly, “Distributed connection acceptance control for a connectionless network”, *Proc. Internat. Teletraffic Congress 16*, 1999, pp. 941-952.
9. R.J. Gibbens, F.P. Kelly, “Resource pricing and the evolution of congestion control”, *Automatica*, vol. 35, pp. 1969-1985, 1999.
10. S. Jamin, S. Shenker, “Measurement-based admission control algorithms for controlled-load service: a structural examination”, University of Michigan, 1997, Technical Report, CSE-TR-333-97.
11. B. Tsybakov, N.D. Georganas, “Overflow probability in an ATM queue with self-similar input traffic”, *Proc. IEEE Internat. Conf. Commun. (ICC’97)*, Montreal, Canada, 1997.
12. A.K. Parekh, R.G. Gallager, “A generalized processor sharing approach to flow control in integrated services networks: The single node case”, *IEEE/ACM Trans. Networking*, vol. 1, pp. 344-357, 1993.
13. R. Bolla, F. Davoli, M. Marchese, “Bandwidth allocation and admission control in ATM networks with service separation”, *IEEE Commun. Mag.*, vol. 35, no. 5, pp. 130-137, May 1997.
14. J.P. Cosmas, G.H. Petit, R. Lehnert, C. Blondia, K. Kontovassilis, O. Casals, T. Theimer, “A review of voice, data and video traffic models for ATM”, *Europ. Trans. Telecommun.*, vol. 5, no. 2, pp. 11-26, March-April 1994.
15. R. Guérin, H. Ahmadi, M. Naghshineh, “Equivalent capacity and its application to bandwidth allocation in high speed networks”, *IEEE J. Select. Areas Commun.*, vol. 9, no. 7, pp. 968-981, Sept. 1991.
16. S. Ghani, M. Schwartz, “A decomposition approximation for the analysis of voice/data integration”, *IEEE Trans. Commun.*, vol. 42, pp. 2441-2452, July 1994.
17. R. Bolla, F. Davoli, S. Ricciardi, “A hierarchical control structure for multimedia access networks”, *Proc. IEEE Internat. Conf. Commun. (ICC ’99)*, Vancouver, Canada, June 1999.
18. R. Bolla, F. Davoli, S. Ricciardi, “Resource allocation strategies for multimedia traffic in an ATM-based PON”, *10th IEEE Tyrrhenian Internat. Workshop on*

Digital Communications, Ischia, Italy, Sept. 1998; in F. De Natale, S. Pupolin, Eds., *Multimedia Communications*, Springer Verlag, 1999, pp. 305-319.
19. R. Bolla, A. Dalal'ah, F. Davoli, M. Marchese, “Dynamic route selection at call set-up level in ATM networks”, in D.D.Kouvatsos, Ed., *ATM Networks Performance Modeling and Analysis*, Chapman & Hall, 1997, pp. 121-140.
20. R. Bolla, F. Davoli, M. Marchese, “Distributed dynamic routing of Virtual Circuits in ATM networks under different admission control and bandwidth allocation policies”, *Internat. J. of Parallel and Distributed Systems and Networks*, vol. 2, no. 4, pp. 225-234, 1999.

An Upper Bound to the Loss Probability in the Multiplexing of Jittered Flows

Marco Listanti[1], Fabio Ricciato[1], Stefano Salsano[2]

[1] INFOCOM dept. – Univ. of Rome “La Sapienza”, v. Eudossiana 18 00184 Roma (Italy)
ricciato@coritel.it
[2] CoRiTeL – v. TorVergata 20 00184 Roma (Italy)
salsano@coritel.it

Abstract. In the context of Diffserv networks some services should be characterized by end-to-end quantitative QoS guarantees. In order to provide such guarantees to single flows (or flow aggregates), the end-to-end analysis of delay and loss performance in a Diffserv domain is needed. The impact of jitter should be considered in the performance analysis at the successive nodes along the path of a flow (flow aggregate). Worst-case analysis is a solution to provide deterministic quantitative guarantees, at the price of very low efficiency. As an alternative this paper proposes a probabilistic approach, aimed at providing statistical quantitative guarantees and achieve higher efficiency. The proposed analytical approach is based on the insertion of a discarding device before the FIFO queue, called “dropper”. The purpose of the dropper is to avoid the analysis of the congestion at the burst level in the queue, allowing for the application of an analytical result derived for packet scale conflicts in the *modulated ND/D/1* queue. Simulations are presented that validate the analytical bound. Finally numerical results are provided to evaluate the efficiency of the bound in an admission control scheme.

1 Introduction

Providing QoS guarantees in a packet switched network means also to be able to keep the end-to-end loss level below a target value. In order to do that, a method is essential to evaluate the loss probability at any network node, or at least to provide an upper bound for it, given a certain characterization of the traffic mix feeding the network. On the basis of such a method, one can build an Admission Control mechanism, centralized or distributed, able to ensure that the end-to-end loss probability along any generic path does not exceed the admissible value.

One major problem when dealing with performance analysis in multistage networks is to take into account the effect of jitter. As the traffic flow originated by the generic *i*–th source crosses several nodes (i.e. multiplexer) through the network, let f_i^k represent the arrival process relevant to this flow at the *k*-th node along his path. If the sources are directly connected to the network f_i^1 characterizes the original *i*-th source flow. In general the arrival process f_i^k (k>1) has different characteristics from f_i^1, because of the jitter introduced by the previous k-1 multiplexing processes. In the literature many methods are proposed to analyze the multiplexing performances in different cases depending on the characterization of the flows sources, i.e. f_i^1 in our notation. Such methods can only be applied to the nodes located at the edge of the network, i.e. before any jitter is introduced onto the flows. The problem of performance analysis in the internal nodes, or equivalently the problem of accounting for the effect of jitter, is faced in the literature in the following ways:

– the problem is neglected basing on the conjecture of "negligible jitter"[1];
– the effect of jitter is taken into account in a deterministic worst-case fashion [4] [5] [6].

The first approach can be applied only in a restricted class of cases, i.e. when the involved delays are "small" (compared to a poissonian scenario of equivalent intensity). The second approach, though extensively addressed in recent papers, in general leads to very low efficiency. In particular it has been shown ([4][5][6]) that the deterministic worst-case "explodes" when the number of multiplexing stages increases, accordingly the achievable efficiency diminishes dramatically.

In this paper an alternative approach is proposed, where the effect of jitter, assumed as non-negligible, is taken into account looking for statistical guarantees instead of deterministic worst case guarantees. The approach is mainly based on an analytical result obtained in [1] and [2] for the *modulated N·D/D/1 queue* and successive manipulation given in [3]. The basic idea is to reduce to the analysis of the per-flow arrival patterns in a sliding time window of opportune length: in the internal of such a window the effect of conflicts on the queue state can be evaluated by means of the above mentioned analytical result. The effect of jitter is taken into account in terms of the maximum packet clumping, which has an impact on the maximum number of arrivals in the considered time window. The effect of conflicts at a time scale larger than the considered window is accounted for by means of a discarding device, called "dropper", which is inserted before the queue. Its function is to filter such large-time-scale conflicts thus isolating the conflicts inside the considered time window. Because of the presence of the dropper the overall multiplexer system is non-work-conserving, therefore it can be assumed that the upper bound to loss probability derived for such a system works for the simple FIFO queue multiplexer too. In this sense the dropper is to be considered simply as a "virtual" device which allows for the analytical derivation of the bound.

The rest of the paper is organized as follows: In section 2 the complete analytical approach is presented. In section 3 the results are compared with simulations. Section 4 discusses the application to a sample case study. Finally conclusions are discussed in section 0 along with some directions for further study.

2 The Analytical Approach

2.1 The Model

Consider the multiplexer model of Fig. 1. It represents a generic network node (multiplexer) Ω in the network. Before reaching node Ω, the traffic flow originated by the generic *i*-th source, denoted by $f_{i,s}$, crosses a certain number of previous multiplexing stages which introduce a variable queuing delay (jitter), whose cumulated maximum value will be denoted by Δ_i. The number of different flows entering the node is *I*. The arrival process at node Ω relevant to *i*-th source will be denoted by f_i.

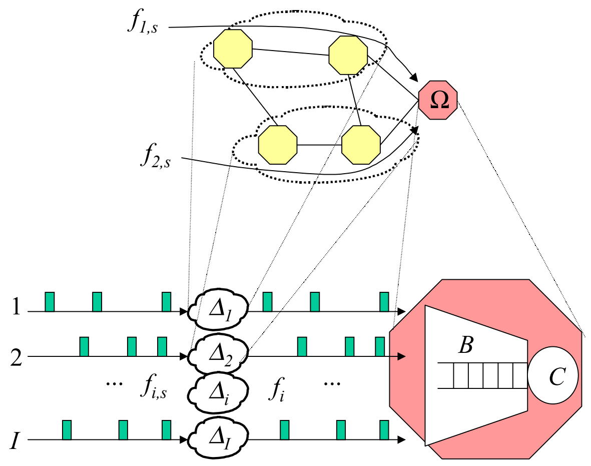


Fig. 1. Network Node Model.

In Fig. 1 *C* and *B* are the output link capacity (bit/s) time and the buffer size (bits) respectively. $\tau=L/C$ will denote the packet service time. The generic source flow $f_{i,s}$ is assumed to be an on/off process, with constant emission rate during the on state and general distribution of active period duration. Fixed packet size is assumed. For $f_{i,s}$ we assume to know:

- $T_{i,s}$: the minimum inter-departure time during the active periods;
- $\overline{T_{i,s}}$: the average inter-departure time in the long term.

The parameters $T_{i,s}$ and $\overline{T_{i,s}}$ equal the inverse of the peak and average packet rates respectively. Given that Δ_i is known (it trivially equals the sum of the queues depletion times along the path from the *i*-th source to node Ω), one can compute the minimum interarrival time for the process f_i:

$$T_i = \max\left(T_{i,s} - \Delta_i, L/C_{in}\right) \quad \textbf{(1)}$$

wherein C_{in} denotes the capacity of the input line to Ω. The average interarrival time for the process is substantially preserved along the flow path, i.e. $\overline{T_i} \cong \overline{T_{i,s}}$, as the average packet rate is affected exclusively by the loss events, whose probability should be kept small (typically not larger than 10^{-4}). One important parameter of the aggregate arrival process to Ω is the minimum interarrival time between consecutive packets of the same flow, which will denoted by $D = min_i\{T_i\}$. Another parameter of interest is the maximum number of arrivals to node Ω in a generic interval of duration W from flow f_i, denoted by $A_i(W)$: it can be easily computed from the source parameter $T_{i,s}$ (and eventually from the maximum burst size in case of sources constrained by a Token Bucket (b,p,r)) and the maximum cumulated queuing delay Δ_i. As an example, here we give the expressions of $A_i(W)$ for a generic on/off source with unlimited maximum burst size (e.g. markovian on/off):

$$A_i(W) = \max\{k : k \cdot T_i - \Delta_i < W\} = \lceil (W + \Delta_i)/T_i \rceil \tag{2}$$

and for an extremal on/off token-bucket-constrained source for the case $W<bp/(p-r)r$:

$$A_i(W) = \min\{\lceil (W + \Delta_i)/T_i \rceil, MBS / L\} \tag{3}$$

In the rest of the paper our aim will be to find an upper bound for the loss probabilities π_i $(i=1,..I)$ and π_{tot} defined as follows:

– π_i is the per-flow loss probability for the generic flow f_i, i.e. the probability that the system can not accommodate a new packet at time t *conditioned* to an arrival at time t from the i-th flow.
– π_{tot} is the loss probability at node Ω for the aggregate π_{tot} as a whole, i.e. the probability that the system can not accommodate a new packet at time t *conditioned* to an arrival at time t (without distinguishing the flow originating the arrival).

In general π_{tot} equals the average of the $\{\pi_i\}$ weighted over the flow average rates R_i, formally ($R = \Sigma_i R_i$):

$$\pi_{tot} = \frac{\sum_{i=1}^{I} R_i \cdot \pi_i}{R} \tag{4}$$

2.2 Case I: Direct Analysis

Under the following condition, which relates the number of different flows with the minimum interarrival time:

$$I \le D/\tau = min_i\{T_i\}/\tau \tag{5}$$

the formula derived in [1 pp. 397÷404] and [2] for the *modulated N·D/D/1 queue* can be directly applied to derive the following upper bound for the generic π_i for the system under study (with a finite buffer of size B and output capacity C):

$$\pi_i \le \sum_{N=0}^{I} P_D(N)\cdot Q_{D,\tau}^{N}(x)\Big|_{x=B/C} \tag{6}$$

wherein $Q^N_{D,\tau}(x)$ is the probability of having an amount of backlog larger than $x{\cdot}C$ in an infinite buffer at the generic instant *t given* the number of arrivals in $[t\text{-}D,t)$ is N:

$$Q_{D,\tau}^{N}(x) \le \sum_{\frac{x}{\tau}<n\le N} \binom{N}{n}\cdot\left(\frac{n\tau-x}{D}\right)^{n}\cdot\left(1-\frac{n\tau-x}{D}\right)^{N-n}\frac{D-N\tau+x}{D-n\tau+x} \tag{7}$$

wherein $\tau = L / C$ is the packet service time and $P_D(N)$ is the probability to collect N arrivals in $[t\text{-}D,t)$. Denoting by $g_{D,i}(k)$ the probability of k arrivals from flow f_i in $[t\text{-}D,t)$, $P_D(N)$ is given by the convolution of the single $g_{D,i}(k)$ as the flows are independent, formally:

$$P_D(N) = conv\{g_{D,i}(k),\ i=1,..I\} \tag{8}$$

As D is smaller than T_i, the generic $g_{D,i}(k)$ is defined only in $k\in \{0,1\}$, precisely:

$$g_{D,i}(k)=\begin{cases} D/\overline{T_i} & k=1 \\ 1-D/\overline{T_i} & k=0 \\ 0 & \text{otherwise} \end{cases} \tag{9}$$

Unfortunately, condition (5) is quite restrictive, so that the bound (6) is not applicable in a wide range of cases of practical interest. The scope of the following section is to derive an upper bound for the cases where condition (5) is not met.

It is useful to point out that the queue analysis found in [1] and [2] which led to bound (6) is based on the following properties of the overall arrival process with reference to a generic interval θ_D of duration D:

i. at most one arrival from each flow in θ_D;
ii. the arrivals are randomly distributed in θ_D;
iii. the maximum number of arrivals in θ_D is not larger than the number of packets that can be served in the same interval.

Note that:

- property *i)* derives from the definition of D;
- property *ii)* derives from *i)* and from the independence of the flows;
- property *iii)* derives from *i)* and (5).

Finally, from (4) it derives that the right-hand term of (6) is also an upper bound for the aggregate loss probability π_{tot}.

2.3 Case II: The Analysis in Presence of a Dropper with Short Dropping Window

Condition (5) can heavily limit the number of flows that can be multiplexed. Condition (5) implicitly implies that $D > \tau$. Now we face the case where for the input traffic aggregate the condition $D > \tau$ still holds but the number of multiplexed flows is larger than D/τ, i.e. condition (5) is not met. In this case, by inserting a device called dropper$_W$ before the FIFO queue, we can refer to the system in Fig. 2, denoted by $S(W,\{f_i\})$.

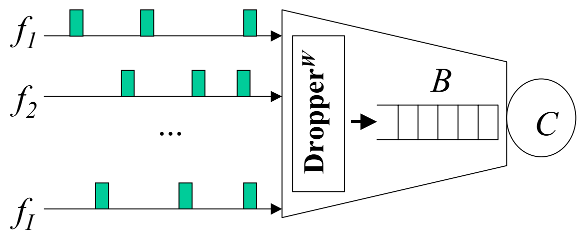


Fig. 2. The system $S(W,\{f_i\})$.

The dropper$_W$ admits a packet arriving at time t if and only if the number of packets admitted in $[t\text{-}W,t)$ is smaller than W/τ-1, otherwise it discards the packet. The duration of the dropping window is indicated as a subscript in "dropper$_W$" to stress that such a parameter characterizes the device. Under the following condition:

$$W \le D = min_i\{T_i\} \qquad \textbf{(10)}$$

the following single upper bound can be given for the generic π_i ($i = 1,..I$) in the system $S(W,\{f_i\})|_{W\le D}$:

$$\pi_i \le \sum_{N=0}^{N_d-1} P_W(N)\cdot Q_{W,\tau}^N(x)\Big|_{x=B/C} + \sum_{N=N_d}^{I} P_W(N) \qquad \textbf{(11)}$$

wherein $N_d = \lfloor W/\tau \rfloor$ while $P_W(N)$ and $Q^N{}_{W,\tau}(x)$ are given by equations (7) through (9) by substituting W to D. This result is equivalent to that found in [3]. Demonstration of (11) is given in *Appendix A*. Substantially, the first and second terms in (11) account for the loss probability at the queue and at the dropper$_W$ respectively. It can be seen that the term relevant to the loss due to the queue is similar to (6). In facts the queue analysis for the two cases is substantially the same, as also in case II the arrival process *to the queue* holds the same properties *i)* through *iii)* defined above with reference to a generic interval θ_W of duration W. As a difference with the case I, in case II property *iii)* is a consequence of the dropper action - rather than of condition (5). It should be clear that the presence of the dropper$_W$ along with condition (10) have replaced condition (5). Finally, it can be noted that when $W = D \ge I\cdot\tau$ the bounds (11) and (6) are equivalent.

The meaning of the presence of dropper$_W$ will be discussed in section 2.5.

2.4 Case III: The Analysis in Presence of a Dropper with Long Dropping Window

The aim of this section is to provide an upper bound for the loss probability in the system of Fig. 2 (i.e. in presence of the dropper$_W$) in case condition (10) is not met, i.e. the dropping window length is longer than $D = min_i\{T_i\}$. This system will be denoted by $S(W,\{f_i\})|_{W>D}$: In this case it is not possible in general to identify any reference interval for which the above mentioned properties *i)* and *ii)* hold. In facts in any time window θ_W of duration W it is possible to collect more than one arrival from the same j-th flow, provided that $T_j < W$. That has the following consequences:

a) the distribution $g_{W,j}(k)$ of the number of arrivals from the j-th flow in the generic interval $[t-W,t)$ is unknown; in facts in general $g_{W,j}(k)$ >0 also outside the set $k \in \{0,1\}$, then from $E\{g_{W,j}(k)\}$ only is not possible to univocally identify $g_{W,j}(k)$ for any k.
b) the arrivals in the generic interval $[t-W,t)$ are not uniformly distributed, as the epochs of arrivals from the same flow are not independent.

From a) it derives that the distribution $P_W(N)$ of the arrivals in $[t-W,t)$ can not be computed, while from b) it derives that the expression (7) for $Q^N_{W,\tau}(x)$ can not be applied. In order to find an upper bound for the generic π_i in $S(W,\{f_i\})|_{W>D}$, we will build a virtual scenario S_i^* such that:

- in S_i^* the per-flow loss probability for i-th flow π_i^* can be analytically upper bounded;
- S_i^* is "worse" than $S(W,\{f_i\})|_{W>D}$ in terms of the per-flow loss probability for i-th flow i.e. $\pi_i \le \pi_i^*$.

In this strategy, building up a "worse" system S_i^* is somewhat critical: roughly, our proposal is to consider S_i^* to be derived from $S(W,\{f_i\})|_{W>D}$ by substituting a "worse" flow f_j^* to each actual flow f_j ($1 \le j \le I$, $j \ne i$), and assuming a "worst case" arrival pattern for f_i in $[t-W,t)$. Denote by m_j ($1 \le j \le I$) the maximum number of arrivals that can be collected in $[t-W,t)$ from flow f_j, and by $M = max_j\{m_j\}$. When focusing on the loss probability π_i for a packet arriving at epoch t from flow f_i we will build the worse system S_i^* in the following way:

1. replace each flow f_j ($1 \le j \le I$, $j \ne i$) with a "worse" flow $f_j^*(M)$ with the same average rate in the long term but for which arrivals occur only in batches of size M packets (it is equivalent to assume packets of fixed size $M \cdot L$)
2. similarly, replace flow f_i with a "worse" flow $f_i^*(m_i)$ with the same average rate but for which arrivals occur in batches of size m_i.

The system S_i^* can be ideally realized in the way depicted in Fig. 3: before entering the multiplexer, each packet flow passes a "re-packing" stage, of parameter M for f_j ($1 \le j \le I$, $j \ne i$) and m_i for f_i. The "re-packing" stage of size k (k positive integer) has the function to gather the packets of a single flow in batches of size k: it buffers the input packets until it collects k ones, then sends them consecutively in a single batch. As $k \ge 1$, the "re-packing" stage concentrates the activity periods of the flows. Considering that the single packet *flows are randomly phased*, *in the average* the system S_i^* will present loss probabilities greater than $S(W,\{f_i\})|_{W>D}$, in other words it is statistically "worse" in terms of loss. System S_i^* is similar to the system considered in section 2.3 (case II) as all the batch arrivals in $[t-W,t)$ belong to different flows, thus are independent. Our approach will be to consider each batch of M arrivals as the arrival of a single packet with service time $\tau{\cdot}M$ in order to apply (7).

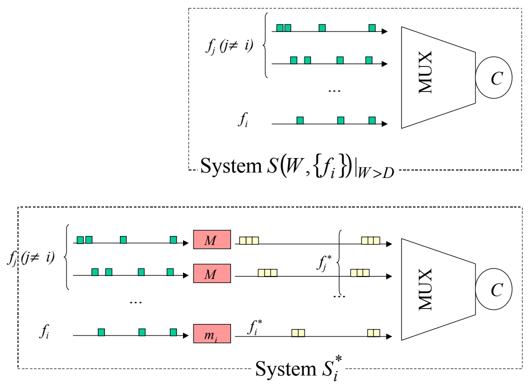


Fig. 3. System with batch arrivals.

Let's cluster the set of flows f_j according to the values of m_i, by defining the *h-class* F_h ($1 \leq h \leq M$) as the set of flows for which $m_i = h$, i.e. $F_h=\{f_i: m_i=h\}$. Following an approach similar to (11), for any flow of class F_h the following bound can be derived:

$$\pi_i \leq \sum_{k=0}^{N_d^M-1} P_W^*(k)\cdot Q_{W,\tau\cdot M}^k(x)\Big|_{x=\frac{(B+L-hL)}{C}} + \sum_{k=N_d^M}^{I} P_W^*(k) = \Pi_W(h) \tag{12}$$

wherein $N_d^M = \lfloor (N_d - h)/M \rfloor + 1$ and

$$P^*_W(k) = conv\{g^*_{W,j}(k), j=1,..I\} \tag{13}$$

$$g^*_{W,j}(k) = \begin{cases} W/(\overline{T_j}\cdot M) & k=1 \\ 1-W/(\overline{T_j}\cdot M) & k=0 \\ 0 & \text{otherwise} \end{cases} \tag{14}$$

Demonstration is given in appendix B. Note that (12) includes (11) as a particular case when M = 1. Similarly to (11) the first and second terms in (12) account for the loss at the queue and at the dropper$_W$ respectively. The multiplicity of bounds expressed by (12) is equal to the number of different classes. Our aim is now to derive a single bound for the loss probability π_{tot} relevant to the input aggregate as a whole. Having in mind equation (4) the following bounds can be derived from (12):

1. averaging (12):

$$\pi_{tot} \le \frac{\sum_{h\in H} R_h \cdot \Pi_W(h)}{R} \tag{15}$$

wherein R_h represents the sum of the mean rates for all the flows in the class F_h and H represents the set of values for h.

2. finding the maximum of (12):

$$\pi_{tot} \le \max_{h\in H}(\Pi_W(h)) = \Pi_W(h)\big|_{h=M} \tag{16}$$

The bound (15) is tighter than (16), although the last one is more immediate to compute. Note that for the homogeneous scenario where only one class is present ($m_i = M\ \forall i$) the two bounds are equivalent.

2.5 Implementation Aspects

As the right-hand term in (6) is not dependent on i, it can be shown that it is an upper bound for the aggregate loss probability π_{tot} as well as for each single π_i ($i=1,..I$). The same applies to bound (11). Thus, for the cases I and II upper bounds for π_{tot} are available. Additionally, (15) and (16) provide bounds for π_{tot} for the case III. So far we have explored the cases II and III by assuming that a dropper acting on a time window of length W were present before the queue (Fig. 4b).

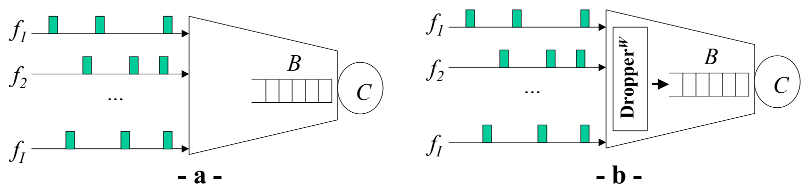


Fig. 4. Reference systems.

It must be noted that the multiplexing system of Fig. 4b is *not* work-conserving because of the presence of the dropper$_W$. As a consequence, the aggregate loss for such a multiplexer fed with a input traffic mix $\{f_i\}$, denoted by $\pi_{tot}^{(b)}$, can *not* be smaller than the aggregate loss for a multiplexer simply constituted by the FIFO queue (Fig. 4a) fed with the same input traffic mix, denoted by $\pi_{tot}^{(a)}$. Formally:

$$\pi_{tot}^{(a)} \le \pi_{tot}^{(b)} \tag{17}$$

As a consequence, all the bounds for the aggregate loss found above for the cases II and III in presence of a dropper$_W$ work also for the simple FIFO queue scheme. Unfortunately, the same does *not* apply for the single *per flow* loss, i.e. in general one can *not* be sure that $\pi_i^{(a)} \le \pi_i^{(b)}\ \forall i$.

In a real scenario, one can choose not to implement the dropper device: it only acts as a "virtual device" useful to derive the bounds for the *aggregate* loss probability. In this context the duration W of the dropper window could be arbitrarily chosen in order to minimize the bound to the loss probability. In particular, given $D = min_i\{T_i\}$ if $I \leq D$ (i.e. condition (5) holds), (6) can be used to evaluate the bound to the loss probability and there is no need to use the dropper and to optimize W. Conversely, if $I > D$ one should optimize the choice of W to obtain the tightest upper bound. The effect of choosing different values of W is depicted in Fig. 5. This figure plots the bound to the loss probability evaluated using (12). The two components of the bound relevant to the dropper and to the queue are represented. The bound is evaluated for a homogeneous scenario where a set of 120 sources is multiplexed at a generic stage (parameters are shown in Tab. 1), and the cumulative maximum delay which has been encountered by each source is $\Delta=50$ *ms*. For this case $T_{i,s} = 72$ ms, while $D = min_i\{T_i\} = T_{i,s} - \Delta = 22$ *ms*. Increasing W, the discontinuities in the curve occur when the maximum number of arrivals m that can be collected in the window of duration W increases. The minimum bound of the loss probability is achieved for $W \cong 94$ *ms*.

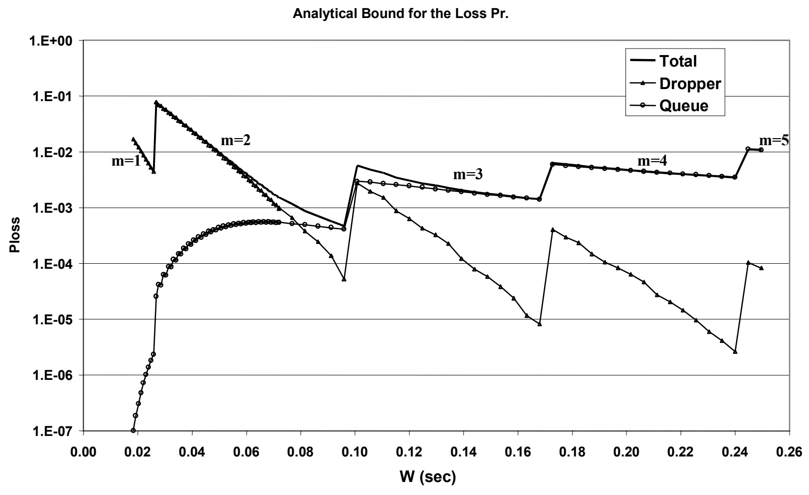


Fig. 5. Analytical bound vs. W (Δ=50 ms).

As an alternative to the approach of considering the dropper$_W$ as a virtual device, one can choose to actually implement it in the multiplexer. The *cons* are that for each dropper the dropping window duration, i.e. the parameter W, must be somehow managed, either statically or dynamically according to the mix of input flows. The *pros* of such a choice is that the bounds found previously for the *per flow* loss probabilities can be exploited *to guarantee a target maximum loss to the single flows rather than simply to the aggregate as a whole*. This would be a major advantage from a theoretical point of view, although in practice it is often sufficient to be able to control the aggregate loss.

3 Simulations and Numerical Results

Simulations have been run in order to validate the upper bounds provided above. Here for sake of simplicity we present exclusively the results relevant to the bound of case III (eq. (12)). Further simulations relevant to case II are available in [3].

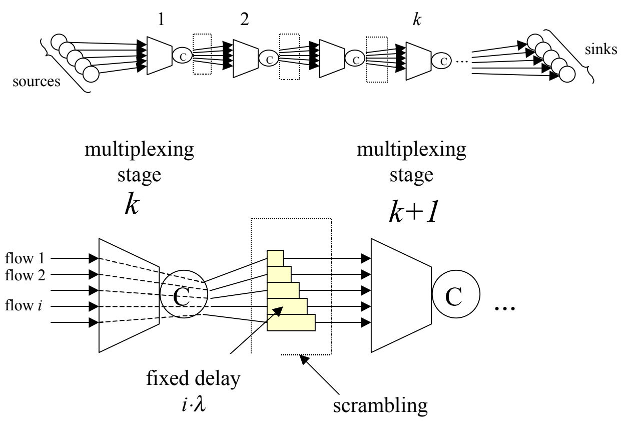


Fig. 6. Simulation scenario

The simulation scenario consists of a linear chain of identical multiplexing stages with the same output capacity and buffer size (see Tab. 1). A set of identical and independent packet sources with is attached directly to the 1st multiplexing stage. Before entering the *k+1* (*k=1,2..*) multiplexing stage, the flows are scrambled in order to eliminate the correlation in time introduced by the previous *k* stages: a fixed delay $i\cdot\lambda$ is added to the packets of the generic *i*-th flow, with λ large compared to the average burst duration. This way the flows in input to the generic multiplexing stage *k* present identical statistical properties as *i)* were originated by identical sources and *ii)* experienced identical jitter processes as passed through the same previous multiplexing processes. This simulation technique was also used in [7].

Table 1. Simulation parameters

Link capacity $C = 5\ Mb/s$	Peak rate $P = 64\ Kb/s$
Buffer size $B = 10\cdot L$	Activity $a = 0.4$
Packet size $L = 576bytes$	$T_{on} = 1.2\ sec$, $T_{off} = 1.8\ sec$

Two kinds of on/off sources were considered (parameters are shown in Tab. 1):

- **Markovian sources**: with *average* duration of active and idle period T_{on} and T_{off} respectively;
- **Extremal TB sources**: with active and idle periods of *fixed* duration T_{on} and T_{off} respectively: such sources are representative of extremal on/off sources con-

strained by a token bucket with parameters p = 64 Kb/s, r = 25.6 Kb/s, b = 5760 bytes.

All the simulations confirmed that the empirical loss stays below the analytical bound at each multiplexing stage. For sake of simplicity only the curves relevant to 3rd multiplexing stage in the case of the Extremal TB sources are presented. Two sets of simulations have been run: in the first set the dropper$_W$ device was actually implemented at each stage, i.e. the multiplexer scheme was that of Fig. 4b. The dropping window duration was set at W = 84 ms, which resulted in a value of $m_i = 2\ \forall i$ at the 3rd stage, as can be derived from eq (3) considering that the maximum queuing delay introduced by the two previous multiplexing stages is $\Delta = \Delta_i = 18.4$ ms $\forall i$ and the minimum inter-departure time is $T = T_i = 72$ ms $\forall i$. Fig. 7 shows the empirical percentage of packet lost in the whole multiplexer with the analytical bound given by (12) (only one class is present) for different loads. The load was varied by varying the number I of multiplexed flows. Note that the abscissa axis reports the *average* load, i.e. $\rho_{av} = I{\cdot}P{\cdot}a/C$. Fig. 7 also compares the empirical percentage of packets lost at the *queue* and at the *dropper* respectively with the first and second terms of (12).

Fig. 7 clearly shows that the bound is met in the whole range of considered loads. Moreover it is evident that the analytical curves follow quite well the behavior of the empirical ones.

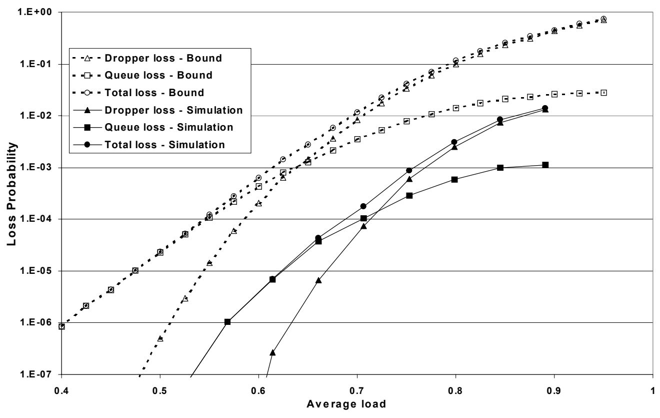


Fig. 7. Comparison between bounds and simulation (1)

In the second set of simulations the dropper$_W$ was not implemented at all, i.e. the multiplexer scheme was the simple FIFO queue of Fig. 4a. Fig. 8 shows the empirical percentage of packet lost vs. the average load, and compares them with analytical bounds computed for different values of parameter W. It can be seen that the empirical curve remains below the analytical bounds for all the considered values of parameter W.

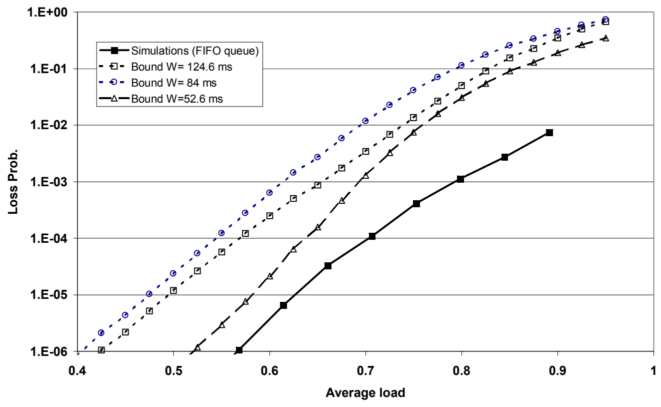


Fig. 8. Comparison between bounds and simulation (2)

4 Application to a Case Study

Throughout this paper we derived upper bounds for the loss probability at a generic network node. The effect of jitter introduced by previous multiplexing stages, supposed non negligible, has been properly taken into account. Such bounds can be used to perform Admission Control, in order to provide a service with loss guarantees in a Diffserv network. In order to gain an insight about the efficiency of an Admission Control scheme based on our bounds, we computed the achievable utilization efficiency for the following sample case. Consider a cascade of multiplexers with output capacity $C = 10\ Mb/s$ and buffer size $B = 30$ packets, fed by on/off packet flows with packet size $L = 576$ bytes, peak rate $P = 128\ Kb/s$ and activity $a = 0.7$. Denote by Ω_{K+1} the generic $(K+1)$-th multiplexer along the cascade: the maximum variable delay (jitter) introduced by the previous multiplexing stages onto the flows entering Ω_{K+1} is $\Delta_{K+1} = K \cdot B \cdot L/C$. Given a maximum admitted loss probability $\Pi = 10^{-4}$, we computed the maximum number of flows that can be admitted to cross Ω_{K+1} so that the bound (16) stays below the threshold Π. Tab. 2 shows the results for different values of K, along with the resulting *average* utilization efficiency $\rho_{av} = I_{max} \cdot P \cdot a/C$.

Table 2. Maximum number of flows and achievable average load for a sample case ($\Pi = 10^{-4}$).

K	1	2	3	4	5	6	7
I_{max}	82	74	72	66	59	58	52
ρ_{av}	0.73	0.66	0.65	0.59	0.53	0.52	0.47

It can be seen from Tab.2 that the achievable efficiency decreases with the multiplexing stage due to the effect of jitter, but the decrease is not dramatic as for the worst-case based admission control schemes [4], which present much lower efficiency. The values of achievable efficiency provided in Tab. 2 suggest that the proposed bounds are feasible to be used in Admission Control schemes of practical interest.

5 Conclusions and Future Works

We have presented a statistical approach to bound the loss probability in a network multiplexer fed by jittered flows. A set of simulations have been run in order to validate the bounds. Some numerical results have been presented, which show that such bounds can be used to perform Admission Control in a more effective way than by relying on deterministic worst-case approaches.

As directions for further study we envisage extensions to the proposed approach in order to *i*) deal with variable size packets and *ii*) take into account the effect of priority scheduling.

Appendix A

We want to derive an upper bound for π_i, i.e. the probability that the system is not able to accommodate (i.e. discards) an arrival at time *t conditioned* to the event Φ_i = "one packet arriving from flow f_i at epoch *t*". Consider that a packet arriving at *t* can be discarded either at the dropper$_W$ if "the number of packets already admitted in $[t-W,t)$ equals $N_d = \lfloor W/\tau \rfloor$" (denote such an event by Θ_d) *or* at the queue if "the backlog in t^- is larger than *B*" (denote such an event by Θ_q)[1]. Using this formalism we can write $\pi_i = Pr\{\Theta_d \cup \Theta_q \mid \Phi_i\}$. By partitioning on the number of arrivals in $[t-W,t)$, denoted by A_W, we can write:

$$\pi_i = \Pr\{\Theta_d \cup \Theta_q \mid \Phi_i\} = \sum_N \Pr\{\Theta_d \cup \Theta_q \mid \Phi_i, A_W = N\} \cdot \Pr\{A_W = N \mid \Phi_i\} \le \qquad \textbf{(18)}$$
$$\le \sum_N \Pr\{\Theta_d \cup \Theta_q \mid \Phi_i, A_W = N\} \cdot \Pr{}_W(N)$$

with $P_W(N) = Pr\{A_W = N\}$. The last inequality holds as the arrival at epoch *t* from flow f_i excludes the possibility to collect one arrival from *c* itself in $[t-W,t)$ as $W \le T_i$, while the arrivals from the other flows f_j ($j \ne i$) are independent from Φ_i: then it derives that substituting $Pr\{A_W = N\}$ in place of $Pr\{A_W = N \mid \Phi_i\}$ leads to an upper bound. Further split the sum in (18) into two, for $0 \le N \le N_d-1$ and $N_d \le N \le I$, and consider that the event Θ_d has null probability when $A_W < N_d$. We end up with the following inequality:

$$\pi_i \le \sum_{N=O}^{N_d-1} \Pr\{\Theta_q \mid \Phi_i, A_W = N\} P_W(N) + \sum_{N=N_d}^{I} \Pr\{\Theta_d \cup \Theta_q \mid \Phi_i, A_W = N\} P_W(N) \qquad \textbf{(19)}$$

As the arrival at time *t* does not influence the queue state at t^- the term $\Pr\{\Theta_q \mid \Phi_i, A_W = N\}$ in the first sum can be rewritten as $\Pr\{\Theta_q \mid A_W = N\}$: it represents the probability to have an amount of backlog larger than *B* at *t conditioned* to *N* ($N < W \cdot \tau$) arrivals in $[t-W,t)$ for a finite buffer system, and can be bounded by the same probability for the case of infinite buffer, i.e. $Q^N_{W,\tau}(x)|_{x=B/C}$ as given by eq. (7) with *W* in place of *D*. Note that eq. (7) can be applied as the arrivals in $[t-W,t)$ are

[11] Note that a service system with fixed packet size *L* and a buffer of size *B* can contain up to *B+L* work: *B* in the queue plus *L* in the servent.

independent (they all belong to different flows) and then uniformly distributed in [*t*-*W,t*). Finally, the term $\Pr\{\Theta_d \cup \Theta_q \mid \Phi_i, A_W = N\}$ in the second sum can be roughly bounded by 1. From (19) we can thus derive (11).

Appendix B

Let's consider a flow f_i belonging to the class F_h, which means that the maximum number of arrivals that can be collected in [*t*-*W,t*) from flow f_j is $m_i = h$. As we have no information about the pattern of such arrivals, i.e. the correlation between the arrival epochs, we consider the "worst case" scenario in which such correlation is maximal, i.e. the *h* arrivals occur at the same epoch *t*. Denote by Φ_i^h the event "one batch of size *h* arrives from flow f_i at epoch *t*". The loss probability π_i can be expressed as follows:

$$\pi_i = \sum_{r=0}^{h} \frac{r}{h} \cdot P_r \le \sum_{r=1}^{h} P_r \tag{20}$$

wherein P_r represent the probability that *r* packets are discarded over the *h* ones constituting the batch arrival from f_i. Equality holds for $h = 1$. In a fashion similar to Appendix A, denote by Θ_d^h the event "the number of packets already admitted in [*t*-*W,t*) equals $N_d - h$" (where $N_d = \lfloor W / \tau \rfloor$) and by Θ_q^h the event "the backlog in t^- is larger than $B + L - h \cdot L$", i.e. the buffer can not accommodate all the incoming *h* packets. Clearly the sum in the last term of (20) represents the probability that *at least one* packet is discarded - either at the dropper or at the queue - *conditioned* to event Φ_i^h, then using the formalism introduced above we can write from (20):

$$\pi_i \le \Pr\{\Theta_d^h \cup \Theta_q^h \mid \Phi_i^h\} \tag{21}$$

Following the same procedure used for (18) in Appendix A, (20) can be partitioned on the number A^*_W of arrivals in [*t*-*W,t*). Note that in the system under study the number of arrivals in [*t*-*W,t*) is an integer multiple of *M*, i.e. $A^*_W = k \cdot M$ with *k* integer. Then we obtain:

$$\begin{aligned}\pi_i &\le \sum_{k=O}^{I} \Pr\{\Theta_d^h \cup \Theta_q^h \mid \Phi_i^h, A_W^* = k \cdot M\} \cdot \Pr\{A_W^* = k \cdot M \mid \Phi_i^h\} \le \\ &\le \sum_{k=O}^{I} \Pr\{\Theta_d^h \cup \Theta_q^h \mid \Phi_i^h, A_W^* = k \cdot M\} \cdot \Pr\{A_W^* = k \cdot M\}\end{aligned} \tag{22}$$

Further split the sum into the ranges where $0 \le A^*_W \le N_d - h$ (or equivalently $0 \le k \le (N_d - h)/M$) and $N_d - h + 1 \le A^*_W \le I \cdot M$ (or equivalently $(N_d - h)/M < k \le I$). Considering that the event Θ_d^h has null probability when $A_W \le N_d - h$, we end up with the following equation (similar to (19)):

$$\pi_i \le \sum_{k=O}^{N_d^M - 1} \Pr\{\Theta_q^h \mid \Phi_i, A_W^* = k \cdot M\} P_W^*(k) + \sum_{k=N_d^M}^{I} \Pr\{\Theta_d^h \cup \Theta_q^h \mid \Phi_i, A_W^* = k \cdot M\} P_W^*(k) \tag{23}$$

wherein $N_d^M = \lfloor (N_d - h)/M \rfloor + 1$ and $P^*_W(k)$ has replaced $\Pr\{A^*_W = k \cdot M\}$. Like for (19) the term $\Pr\{\Theta_q^h \mid \Phi_i, A_W^* = k \cdot M\} = \Pr\{\Theta_q^h \mid A_W^* = k \cdot M\}$ in the first sum represents the probability to have an amount of backlog larger than $B+ L - h \cdot L$ at t^- *conditioned* to k arrivals (of batches with M packets) in $[t-W,t)$ for a finite buffer system. To find an upper bound for it, we can apply eq. (7) by considering that a batch of M arrivals is equivalent to the arrival of a single packet with service time $\tau \cdot M$ and that different batch arrivals belongs to different flows and thus are independent. Formally we have for $0 \le k \le N_d^M - 1$:

$$\Pr\{\Theta_q^h \mid \Phi_i, A_W^* = k \cdot M\} \le Q_{W,\tau \cdot M}^k(x) \Big|_{x=\frac{(B+L-hL)}{C}} \qquad \mathbf{(24)}$$

Finally, the term $\Pr\{\Theta_d^h \cup \Theta_q^h \mid \Phi_i, A_W^* = k \cdot M\}$ in the second sum can be roughly bounded by 1. Thus from (24) we derive (12).

References

[1] U. Mocci, J. Roberts, J. Virtamo eds., “Broadband Network Teletraffic”, Final report of action Cost-242, Springer 1996.

[2] I. Norros, J. W. Roberts, A. Simonian, J. T. Virtamo, “The Superposition of Variable Bit Rate Sources in an ATM Multiplexer”, IEEE JSACApril 1991, pp. 378-387.

[3] M. Listanti, F. Ricciato, S. Salsano, “Delivering End-to-end Statistical QoS Guarantees for *Expedited Forwarding*”, SPECTS 2000, Vancouver.

[4] M.Listanti, F.Ricciato, S.Salsano, L.Veltri “Worst-Case Analysis for Deterministic Allocation in a Differentiated Services Network" To appear in IEEE Globecom 2000, S. Francisco, USA, November 2000.

[5] A. Charny "Delay bounds in a network with aggregate scheduling", Draft version 3, 2/99, Cisco, Available from ftp://ftpeng.cisco.com/acharny/aggregate_delay.ps

[6] J.Y. Le Boudec "A proven delay bound for a network with Aggregate Scheduling" EPFL-DCS Technical Report DSC2000/002, http://ica1www.epfl.ch/PS_files/ds2.pdf .

[7] G. Bianchi, R. Melen, “Effects of multiple nodes crossing on ATM traffic Performance”, European Transaction on Telecommunications, Jan-Feb 1999

SMART: A Scalable Multipath Architecture for Intra-domain QoS Provisioning

Srinivas Vutukury[1] and Jose J. Garcia-Luna-Aceves[2]

[1] Computer Science Department, University of California,
Santa Cruz, CA 94065, USA,
vutukury@cse.ucsc.edu

[2] Computer Engineering Department, University of California,
Santa Cruz, CA 94065, USA,
jj@cse.ucsc.edu

Abstract. The main concern with the IETF proposed Intserv architecture is that the use of per-flow routing and reservation state in the routers may not be scalable to high-speed backbone networks. This paper proposes the Scalable Multipath Aggregated RouTing (SMART) architecture that aggregates flows along multipaths such that the per-flow reservation state in the routers is reduced to a small scalable aggregated state whose size is dependent only on the number of destinations and flow classes. The SMART architecture can be implemented in current IP networks and the complexity is similar to the best-effort IP architecture.

1 Introduction

Several different architectures have been proposed to support applications that require strict service guarantees. In the IETF Integrated Services architecture (Intserv) [3,21] and other architectures [10,12,15], the network reserves the required bandwidth on *per-flow* basis, and ensures that the flows receive their allotted bandwidth by using fair schedulers, such as Weighted Fair Queuing (WFQ) or equivalent [4,13,14], at each link. However, these per-flow approaches have many scaling problems as described below.

Firstly, as the links in backbone networks reach or exceed gigabit capacities, routers are expected to carry large numbers of flows, which requires large amounts of memory to store the routing and reservation state in the routers. Secondly, as the reservation state increases, maintaining the consistency of reservations in the presence of network failures and control message loss becomes complex in per-flow architectures. To maintain the consistency of reservation state, the IETF proposed the soft-state reservation protocol RSVP [21], which though robust can be prohibitively expensive due to its use of per-flow refresh messages. Thirdly, as the number of flows on a link grows, the complexity of link-schedulers (e.g., [13,14]) grows, thus making it more and more difficult to schedule packets in a timely manner. All these scalability problems arise mainly because the per-flow mechanisms in all these architectures are functions of the number of flows.

Scalability problems of the Intserv architecture are well-known and are currently being addressed by several researchers. To make RSVP scalable, there have been many proposals to reduce its refresh message overhead [1,19]. But unfortunately, they are only partial solutions that mitigate the problem rather than solve the problem; the amount of reservation state that needs to be maintained in the routers is still the same. One approach to providing a scalable Intserv solution is to eliminate the per-flow reservation state in the core routers and follow a *stateless* [1] approach similar to Differentiated Services (Diffserv) architecture [5]. The architectures proposed in [15,22] follow this approach for example. In [15], the reservation state is stored in the packets of the flows instead of storing in the routers. The reservation state in the packets is then used by the core routers to estimate the aggregate reservation on the links. There are no explicit refresh messages and thus the problems associated with lost or delayed refresh messages are greatly diminished. However, there is concern regarding bandwidth underutilization resulting from inaccurate estimation of aggregate reservation on the links. Also, the overhead related to state carried in the packets and processing power required for packet processing can limit the scalability and efficiency of this architecture. In [22], a central broker is used to store the reservations of all the flows. This too has serious scalability problems because the central broker not only has to perform path-selection and termination on per-flow basis but also has to process periodic refresh messages from all the flows in the network if a soft-state mechanism is used to maintain the reservation state.

In this paper, we describe a new architecture called SMART (Scalable Multipath Aggregated RouTing), that addresses the scalability problems of the above architectures. The key idea in the SMART architecture is that flows are aggregated along *multipaths* [16,17]. Multipaths are acyclic directed graphs rooted at the destination and are a generalization of single shortest-path routing trees. In our earlier work [16], we described the main concepts in the SMART architecture using a fluid traffic model and discussed the many benefits of using multipaths. This paper extends that work by using the realistic non-fluid traffic model. Flows are aggregated at the ingress router based on class and destination, and the interior or core routers process only aggregated flows. The aggregation is such that, by providing guarantees to the aggregated flow, the guarantees of the individual flows within the aggregate are also guaranteed. The delay variations introduced by link schedulers are removed through shaping of aggregated flows rather than shaping of individual flows. The flow classes defer from previous approaches (e.g., [2,8,9]) in that they satisfy a *closure property*; the delay bounds obtained for a class are independent of the number of flows that form the group. As a result of the use of multipaths and flow classes the routing and reservation state in the SMART architecture is proportional to the number of destinations rather than the number of the end-user flows; consequently, the soft-state reservation protocol used to maintain the reservation state is also scalable. In our prior work, we described such a reservation protocol, called AGREE [17], which maintains

[1] note that stateless means there is no reservation state stored in the routers, but there is still routing state

consistency of aggregated reservation state of the SMART architecture. Also, the link schedulers in the SMART architecture process only aggregated flows which is proportional to the number of active destinations and not the number of individual flows. Thus, the complexity of all the mechanisms in the SMART architecture are function of the a priori known network parameters (number of destinations and classes) which is similar to the Diffserv architecture. This echoes the main reason why the current best-effort IP architecture is so scalable – the routing state is a function of the number of destinations in the network. The main drawback of using multipaths, however, is that packets may arrive out of order at the destination, but this should not be a concern, if deterministic end-to-end delay bounds are also provided concurrently. Typically, real-time applications use a playback time, which means packets need only arrive within a specific time frame, and arrival order of the packets does not matter.

The rest of paper is organized as follows. Section 2 describes the SMART architecture. In Section 3 we derive the end-to-end delay bounds and describe through an example how a QoS network design based on the SMART architecture can be designed. Section 4 concludes the paper.

2 The SMART Architecture

2.1 Overview

The SMART architecture aims at intra-domain QoS and accordingly assumes a network infrastructure consisting of a single autonomous network running an interior-gateway protocol (e.g., OSPF, IS-IS) that computes distances between routers and propagates link-bandwidth information. For simplicity we assume a non-hierarchical network. The edge routers maintain per-flow state and perform per-flow tasks such as packet classification, path selection and signaling, while the core routers maintain aggregated state and perform packet forwarding and link scheduling on per-aggregated-flow basis.

For each destination, a multipath is constructed using the distances computed by the routing protocol. A multipath is an acyclic directed graph with the destination as the sink node and flows of a particular destination are always established along the multipath of that destination. We describe in detail how multipaths are constructed in Section 2.2. Aggregating flows removes isolation between flows which, in general, results in increased burstiness in traffic. In section 2.3, we present a simple technique to aggregate flows that minimizes traffic burstiness and define a small number of aggregated flow classes based on it. Once flows with a particular destination and class are aggregated, they collectively share the bandwidth allocated to that class along the multipath of that destination and are serviced collectively by the routers. Note that the packets of a flow can follow any path in the successor graph in a connection-less fashion; there is no explicit connection maintained on a per-flow basis.

Figure 1 shows the schematic for a router in the SMART architecture. Router i has N^i links and each outgoing link is serviced by a WFQ scheduler or equivalent. Router i uses token buckets to enforce the rates of the Z flows generated

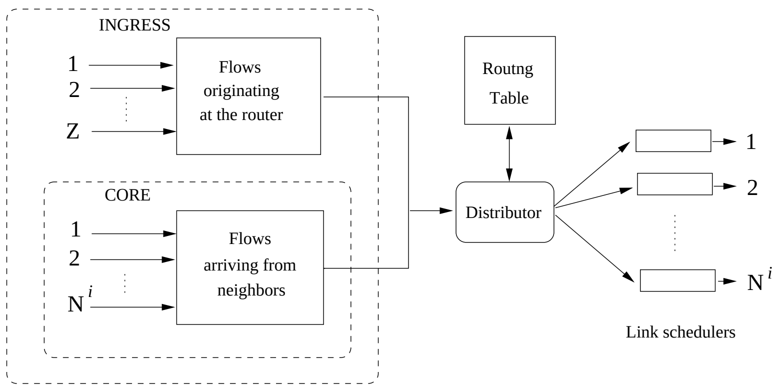


Fig. 1. Block diagram for the SMART router

locally by the host attached to the router. It then encodes the destination and class identifier in each packet before it forwards it the next router. At router i, let S_j^i be the set of next-hop routers for destination j. Packets received by router i destined for router j are only forwarded to neighbors in the set S_j^i. Because there can be more than one next-hop in S_j^i, bandwidth for each of the next-hops must be specified. Let $B_{j,g,k}^i$ specify the aggregated bandwidth of class g and destination j that is forwarded to neighbor $k \in S_j^i$, and let $B_{j,g}^i = \{B_{j,g,k}^i | k \in S_j^i\}$. A routing table entry, therefore, is of the form $\langle j, g, B_{j,g}^i, S_j^i \rangle$, where the class g and the destination j uniquely identify a table entry.

When the router receives a packet with destination j and class g, the corresponding routing table entry is accessed. The first task of the router is to determine a successor k for this packet from S_j^i according to the set $B_{j,g}^i$. This task is performed by the *distributor* (Fig.2(b)), which uses a scheduling discipline to allocate packets to successors according to routing parameter set $B_{j,g}^i$. The algorithm for this has been provided in our prior work [18]. The router then puts the packet in the queue of j and class g at the link scheduler of link to (i, k). The time complexity of determining the next hop by the distributor is constant as there are fixed number of neighbors.

When flows are added and deleted, the routing tables are modified to reflect the bandwidth requirements as follows. Assume a flow request with class g and bandwidth ρ is required to be established between a particular source and destination j. A path P is selected such that, for each link $(i, k) \in P$, $k \in S_j^i$ and $B_{j,g,k}^i + \rho \leq C_k^i$, where C_k^i is the link capacity. The routing table entry is then modified such that $B_{j,g,k}^i \leftarrow B_{j,g,k}^i + \rho$. Similarly, the allocated bandwidth is decremented when flows are terminated. Note that the link admission test takes $O(1)$ time which is much simpler than the admission tests in [8,12,20] which depend on the individual reservations already made to other flows and are generally complex. Also, the link schedulers have to service flows on per-destination

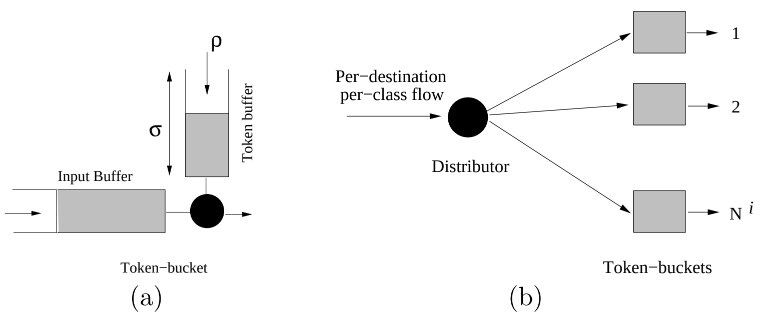


Fig. 2. (a) Dynamic Token Bucket (b) Distributor

per-class basis irrespective of number of flows through the link. And finally, the number of refresh messages on a link used by the soft-state refresh mechanism in AGREE [17] is bounded by $O(NQ)$, where Q is the number of flow classes and N the number of destinations. The rest of this section describes the architecture in further detail.

2.2 Multipaths

Let D^i_j be the distance from router i to router j measured in number of hops. Define the successor set at a router with respect to a destination as consisting of *all neighbors of a router that are closer to the destination, or are at equal distance but with lower address.* The successor set of node i for j is denoted by $S^i_j = \{k | k \in N^i \wedge (D^k_j < D^i_j \vee (D^k_j = D^i_j \wedge k < i))\}$, where N^i is the number of neighbors of router i. Now, with respect to a router j, the successor sets S^i_j define a successor graph $SG_j = \{(m,n) | (m,n) \in E, n \in S^m_j(t),\ m \in N\}$, where N is the set of nodes in the network and E is the set of links. A shortest multipath from router i to j is a generalization of shortest path and is defined as the subgraph of SG_j consisting of all nodes that are reachable from the source i. Fig. 3(b) shows the shortest multipath with destination 0 for the network in Fig. 3(a).

Using the shortest multipath as defined above poses a problem: when two routers are equidistant from the destination, ties are resolved based on addresses, which may result in an unfair and uneven successor graph. Fig. 4(a) shows an example of such an uneven successor graph with 4 as the destination. The longest path from 5 to 4 is four hops and the shortest path is one hop. Since end-to-end delays are determined by the longest path (Eq.(3)), this results in longer delay bounds for flows from E to A. Furthermore, the link (1, 4) is a hot-spot link, because all traffic reaching 1 must be transmitted on link (1, 4) as there is no alternative.

This unevenness can be fixed in a couple of different ways. In the first method, the successor set is restricted to consists of *all neighbors that are strictly closer to the destination than the node itself.* That is, $\hat{S}^i_j = \{k \mid k \in N^i \wedge D^k_j < D^i_j\}$, which

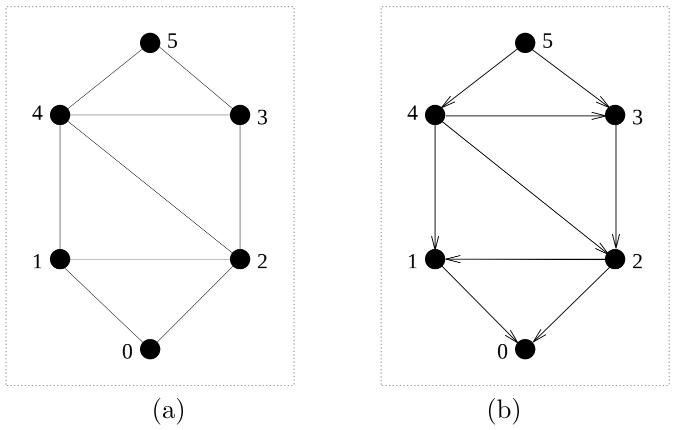


Fig. 3. (a) A sample network (b) Multipath successor graph

is in fact the next-hop set computed by OSPF. A shortcoming of this approach is that it does not use the full connectivity of the network. For example, in Fig. 4(b), which is a successor graph using $\hat{S}^i_j$ for destination 4 in the network given in Fig. 3(a), some links are not used. This results in lower utilization of network bandwidth and higher call-blocking rates.

In the second method, the successor set is defined as $S^i_j = \{k | k \in N^i \wedge D^k_j \leq D^i_j\}$. That is, the successor set with respect to a destination consists of *all neighbors of a router that are closer to or at the same distance from the destination.* We call this the *enhanced multipath* (EMP). For example, Fig. 4(c) shows the EMP for destination 4 in Fig. 3(a). The EMPs fix the unevenness problem of the shortest multipath and also improve the bandwidth utilization over the multipaths computed by OSPF. However, there is one important problem with EMPs that needs to be addressed. Note that each router now includes the neighbor that is equidistant from the destination in its successor set, which can cause packets to loop. This can be easily fixed using the following simple technique. Each packet carries a bit-flag called the *e-bit*, which indicates whether the packet was ever forwarded to a *peer* router (neighbor router that is at the same distant from the destination) on its path so far. A router forwards a packet to a peer neighbor only if the e-bit is set; otherwise, the packet is forwarded to one of the *subordinate* neighbor (neighbor whose distance is strictly less than the distance of this router). If a packet is forwarded to a peer neighbor the e-bit is cleared so that all the future routers visited will forward it only to their subordinate routers. It is easy to see that a packet can be forwarded to peer neighbor at most once, thus preventing packet from looping. The packets of a flow can, therefore, follow at most $(D^i_j + 1)$ hops. The ingress router sets the e-bit of the packets only for those flows that were established along the EMP with length $(D^i_j + 1)$. The per-flow e-bit information of the flows is maintained only at the ingress router. The routing table entries are extended to specify the bandwidths

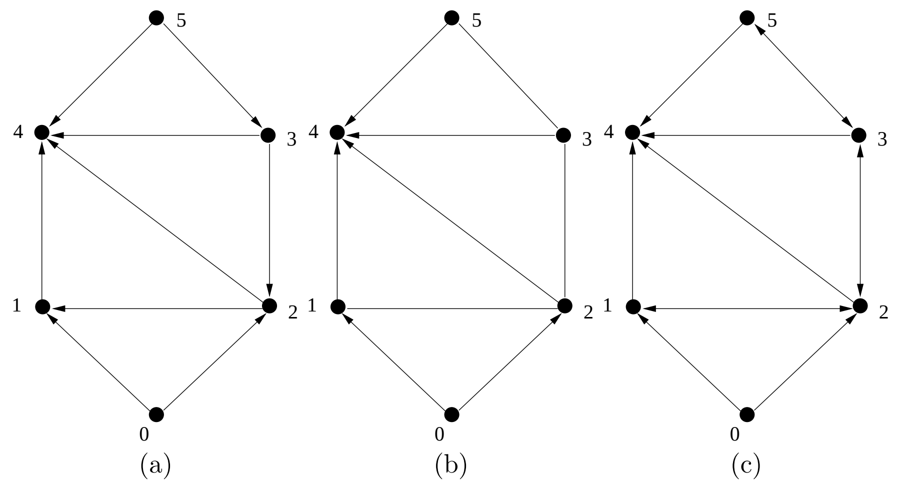


Fig. 4. (a) An uneven successor graph (b) OSPF's multipath (c) enhanced multipath

for traffic with e-bit set and for traffic without e-bit set. The extended routing table entry is $\langle j, g, S^i_j, B^i_{j,g}, \hat{S}^i_j, \hat{B}^i_{j,g}\rangle$. When a packet is received with the e-bit set the distributor uses the $B^i_{j,g}$ to determine the next hop, otherwise the $\hat{B}^i_{j,g}$ is used. As already mentioned, the e-bit is cleared before the packet is forwarded to a peer.

Our intuition for using EMPs follows from the dynamics of Widest-Shortest path-selection strategy which is often used as a benchmark [7]. The Widest-Shortest path-selection algorithm first selects valid paths that are the shortest, and as bandwidth on the shorter paths is consumed, longer paths are tried. Effectively, the requests are first attempted along the EMPs. By always selecting paths along EMPs, bandwidth utilization comparable to Widest-Shortest path can be attained [16].

2.3 Flow Aggregation and Shaping

We characterize a flow by the parameters (L, ρ), where L is the maximum size of any packet of the flow and ρ is the average rate of the flow. Flows are then grouped into classes based on their maximum packet sizes and their rate. Assume there are Q real-time classes and with each class g associate a maximum packet size L_g and rate of ρ_g. A flow belongs to class g if the maximum size of its packets is less than L_g and its rate is at least ρ_g. The flow's class is determined at flow setup and when the application sends its data packets, the ingress router inserts the flow's class identifier in the TOS field of the IP packet header before forwarding it to the network. At the ingress and in the core routers all flows of the same destination and class are merged and handled as a single flow. We now show how, by providing guarantees to the aggregate flow, the guarantees of individual flows in the class aggregate can be ensured.

Flows are shaped at the ingress node to have at most a single packet burst before merging with other flows. That is, a flow f with parameters (L, ρ) is shaped with token bucket with bucket size L and token rate ρ (Fig.2(a)). If the flow f is serviced by a WFQ scheduler at a link (i, k), the delay bound for a packet of the flow d_f is given by

$$d_f = \frac{L}{\rho} + \frac{L_{max}}{C_{ik}} + \tau_{ik}. \tag{1}$$

where τ_{ik} and C_{ik} are the propagation delay and capacity of the link respectively and L_{max} is the maximum packet size allowed for any packet in the network [13]. Now, if the flow belongs to the class g, then $L \leq L_g$ and $\rho \geq \rho_g$ and thus we have $\frac{L}{\rho} \leq \frac{L_g}{\rho_g}$. Therefore,

$$d_f \leq \theta^g_{i,k} = \frac{L_g}{\rho_g} + \frac{L_{max}}{C_{ik}} + \tau_{ik}. \tag{2}$$

That is, $\theta^g_{i,k}$ is the delay bound for the class g at link (i, k). We can easily show that delay bound $\theta^g_{i,k}$ holds for the flow even after it is aggregated with other flows belonging to the same class. Assume that flow f is merged with $n-1$ other flows of the same class g and shaped to a single packet burst. The maximum burst the aggregate flow can have is nL_g. Thus, the resulting aggregated flow can be characterized by the token-bucket parameters (nL_g, ρ^a), where the aggregate bandwidth ρ^a is the sum of rates of the participating flows. The delay bound offered by WFQ to the aggregate is then $\frac{nL_g}{\rho^a} + \frac{L_{max}}{C} + \tau_{ik}$. However, this delay bound cannot be used in the end-to-end delay bound for the flow f, because ρ^a varies in a dynamic environment where flow f may merge with different flows at different times. However, given that $\rho^a \geq n\rho_g$ always holds true, a delay bound of $\frac{nL_g}{n\rho_g} + \frac{L_{max}}{C} + \tau_{ik}$ will always hold for the aggregate flow and the constituent flows. This is of course equal $\theta^g_{i,k}$; therefore, for flow f we can use the delay bound of $\theta^g_{i,k}$ on the link in its computation of end-to-end delay bound irrespective of what flows are aggregated with it and at any time as long as they are of the same class. Note that depending on what flows are part of the aggregation tighter delay bounds can be obtained; however, our objective is to obtain delay bound that is independent of constituent flows.

The link scheduler introduces some delay-jitter in the traffic passing through the link. This is removed at the receiving router by reshaping traffic, again using token buckets, before it is merged with other flows. There is one token-bucket for each class-destination pair at the receiving end of the link. The token buckets are dynamic in the sense that each time a flow is added on the link during flow setup, the parameters of the corresponding token bucket are adjusted. That is, for link (i, k), class g and destination j, at node k, the rate of the corresponding token-bucket is set at rate $B^i_{j,g,k}$ and the bucket-size at $\frac{L_g B^i_{j,g,k}}{\rho_g}$. The maximum delay experienced by a packet of class g in the link scheduler and in the shaper at the receiving end of the link is $\theta^g_{i,k}$. The traffic emerging from the shaper belongs to class g and can be readily merged with the shaped traffic of the same class received on the other links.

Because the distributors cannot split packets, an extra burst is introduced by the distributors, which has to be incorporated in the end-to-end delay bounds. This is removed using a token-bucket shaper as shown in Fig 2(b). There is one token-bucket shaper for each class-destination pair at the output queue of the distributor. The rate of the token-bucket shaper at link (i,k), for j and class g is set at $B^i_{j,g,k}$ and the bucket-size to the $\frac{L_g B^i_{j,g,k}}{\rho_g}$, and is adjusted as flows are setup and terminated. The extra burst can be at most L_g and the proof for this is given in [18]. The delay introduced by the shaper k of the distributor output is $\frac{L_g}{B^i_{j,g,k}}$ which is upper bounded by $\frac{L_g}{\rho_g}$ as $B^i_{j,g,k} \geq \rho_g$.

The number of queues at a link scheduler is of $O(|N|Q)$. This can be reduced to $O(Q)$ by merging all flows of a particular class, irrespective of the destination, into a single queue. Because flows have different destinations, unlike before, the per-destination aggregated flows may have to be extracted from the per-class aggregate at the receiving end of the link. Merely reshaping the class-aggregated flow to fit the class envelope is not sufficient in this case; the class-aggregated flow must be restored completely to the traffic pattern it had before entering the scheduler. To achieve this, instead of using a token-bucket for each class-destination pair at the receiving end of the link, a device called *regulator* [20] is used for each class-aggregated flow. The regulator works as follows. If the delay in link scheduler is θ for a packet, the regulator holds the packet for time $(\theta^g_{i,k} - \theta)$ before forwarding to the distributor. By delaying each packet passing through the link to experience the worst-case delay of $\theta^i_{i,k}$, the regulator restores the class-aggregated flow to the form it had before it entered the link scheduler. Now the packets of a particular destination and class can be extracted from the class-aggregated flow and freely merged with traffic of the same destination and class received on the other links. The disadvantage of this scheme is that a delay field must be used in the packet which makes the scheme difficult to implement in current IPv4 architecture.

Aggregation in the context of guaranteed flows has been discussed before [2,9], but they primarily deal with static aggregation of flows between a source-destination pair, and do not address delay guarantees when individual flows enter and leave the aggregation dynamically. In [8], the delay bounds are dependent on the constituent flows of the aggregated flow. The closure property differentiates our aggregation method from the other aggregation methods. The aggregation scheme proposed for RSVP [6] is meant for aggregating reservation state of flows within a single multicast group. Our aggregation scheme is orthogonal to aggregation of RSVP. In our architecture, aggregated flows are shaped and not individual flows as in [11,12], in which the benefits of per-hop shaping can be realized (e.g., reduction in buffer sizes), but are largely undone by the per-flow traffic management that must simultaneously be employed!

3 End-to-End Delay Bounds

3.1 Multipath Flows

The end-to-end delay bound for flows established along multipaths must include the delays experienced in the distributor and the delays in the link schedulers. Because the links can have different bandwidths, the delay of the worst path in the multipath should be chosen for computing the end-to-end bound. Thus, for a class g flow from router i to j, the end-to-end delay $\delta^i_{j,g}$ is recursively defined as follows

$$\delta^i_{j,g} = \frac{L_g}{\rho_g} + MAX\{\theta^g_{i,k} + \delta^k_{j,g} \mid k \in S^i_j\}. \tag{3}$$

where $\theta^g_{i,k}$ is as in Eq.(2) and $\delta^j_{j,g} = 0$ for all j and g. The first term on the right hand side of Eq.(3) is due to the delay in the distributor. Observe that the end-to-end delay bounds[2] in this architecture can be determined at the ingress node itself using the class and destination of the flow and the link information propagated by the routing protocol. For high-speed backbone networks where the link capacities tend to be very high compared to individual flow bandwidths, the $\frac{L_g}{\rho_g}$ will be the dominant component of Eq.(2). So the $\delta^i_{j,g}$ of Eq.(3) will reduce to

$$\delta^i_{j,g} = (2M^i_j - 1)\theta^g \tag{4}$$

where $\theta^g = \frac{L_g}{\rho_g}$ and M^i_j is the longest path from i to j in the shortest multipath. For an EMP $M^i_j = D^i_j + 1$, and the end-to-end delay-bound is

$$\delta^i_{j,g} = (2D^i_j + 1)\theta^g \tag{5}$$

The error terms ($L_{max}/C_{i,k}$ and $\tau_{i,k}$) due to link capacities and propagation delays can be propagated using the routing protocol. The ingress router simply needs to add this to the end-to-end class delay.

3.2 Single-Path Flows

The delay-bound $\frac{L_g}{\rho_g}$ introduced by the distributor can be expensive for flows that have small bandwidth. The resulting end-to-end delay-bound can be as

[2] note that we assume the end-to-end delay bounds are known a priori; they are not specified by the user, but are provided by the network through design.

much as twice compared to those in per-flow architecture. This, however, can be overcome by simply bypassing the distributor, and forcing all packets of a flow to follow the same path. The flows are still established along multipaths, but flows of the same destination do not share the bandwidth along the multipaths. We have shown how this is implemented in [18]. The packets carry a key which is used to hash into one of the subordinate next-hops. The ingress router maintains the key information in the per-flow table. In the core routers, there is no need for distributing the packets of these classes along the multiple next-hops and thus it eliminates the delay introduced by the distributors, and the delay bound obtained will then be comparable to those provided by per-flow mechanism. If a single-path flow of class g and path P from i to j, then the maximum end-to-end delay-bound $\delta^i_{j,g}$ for this flow is given by

$$\delta^i_{j,g} = \sum_{(m,n)\in P} \theta^g_{m,n} \tag{6}$$

Again, assuming that class delays are the dominating components at all links, we have

$$\delta^i_{j,g} = D^i_j \theta^g \tag{7}$$

Note that the delay bound obtained through Eq.(7) is half that of the delay-bound obtained through Eq.(5). The disadvantage of using single-path flows is that bandwidth utilization can be lower compared to the utilization achieved through multipaths. We explore this effect in a future publication.

3.3 Designing a QoS Network

In this section we discuss how a network supporting QoS can be designed based on the SMART architecture, using an example. The problem can be formulated as follows. We are given (1) a physical network topology, the capacities and propagation delays of the links (2) the characteristics of input flows that the network is expected to support; and (3) the types of delay-bounds that the input flows expect. We want to determine a set of flow classes and their corresponding parameters L_g, ρ_g, and whether the flow class is of single-path or multipath type. The following simple example illustrates the design process. We are given:

1. A network with diameter of 16 hops, link capacities of 150 Mbps and propagation delays of 1 ms.
2. Two types of real-time input flows. Video flows of bandwidths 1-3 Mbps, and audio flows of 64 Kbps.
3. End-to-end delay-bounds of 150 ms for both types of flows. (According to CCITT, this is sufficient to support real-time applications.)
4. Maximum packet size for the system is 1500 bytes. (Using the MTU of Ethernet)

To derive a feasible set of class parameters, set $\rho_{audio} = 64\ Kbps$ and $\rho_{video} = 1\ Mbps$. Using Eq. (7) for audio flows we obtain $L_{audio} \approx 70\ bytes$. For video flows we obtain $L_{video} \approx 620\ bytes$ using Eq.(5).

Observe that the packet sizes are significantly large for video flows while they are quite small for audio flows. This shows that supporting low-bandwidth flows such as audio is generally more difficult than supporting large bandwidth flows such as video, even in the case of per-flow architectures; this is a fundamental problem with low-bandwidth flows. Therefore, it is reasonable to expect low-bandwidth flows to use small packets, if burstiness is to be controlled. Packet sizes can be increased for audio flows, however, if the ingress nodes establish reservations for collection of flows. For instance, the ingress node can aggregate several audio flows, say 10, and setup a single aggregated flow. Because the bandwidth is increased 10 fold, the aggregated flow can belong to a flow-class that has a minimum bandwidth requirement of 640 Kbps, with corresponding maximum packet size of 700 bytes. The resulting larger packet sizes may now be acceptable. Reserving more bandwidth than the flow requires is a technique that is often used to overcome difficulties with supporting low-bandwidth flows.

4 Conclusions

The paper presented a scalable multipath QoS architecture SMART that provides end-to-end delay guarantees to real-time multimedia applications. The architecture uses multipaths to reduce the size of the routing and reservation state to levels comparable to the highly scalable current best-effort IP architecture. The reduced state in the routers can then be managed using the AGREE soft-state reservation protocol [17]. The refresh messages needed in the AGREE architecture is determined by the number of destinations and classes rather than the number of actual flows and is much more scalable than the Intserv/RSVP model. The link schedulers are scalable as they only have to process per-destination per-class aggregated flows.

The main drawback of the SMART architecture is that the delay bounds are loose compared to the tight delays that can be achieved using per-flow processing. We showed through an example how by engineering the parameters of flow classes, acceptable delay bounds for supporting real-time multimedia applications can be obtained.

References

1. L. Berger and et al. RSVP Refresh Overhead Reduction Extensions. *Internet Draft, draft-ietf-rsvp-refresh-reduct-05.txt*, June 2000.
2. S. Berson and S. Vincent. Aggregation of internet integrated services state. *IWQOS*, 1998.
3. R. Braden, D. Clark, and S. Shenker. Integrated services in the internet architecture: An overview. *RFC 1633*, June 1994.
4. A. Demers, S. Keshav, and S. Shenker. Analysis and Simulation of a Fair Queueing Algorithm. *Proc. of ACM SIGCOMM*, pages 1–12, 1989.

5. D. Black et al. An Achitecture for Differentiated Services. *Internet Draft*, May 1998.
6. R. Guerin et al. Aggregating RSVP-based QOS Requests. *Internet Draft*, Nov. 1997.
7. R. Guerin et al. Qos Routing Mechanisms and OSPF Extensions. *Internet Draft*, Jan. 1998.
8. R. Guerin et al. Scalable QoS Provision Through Buffer Management. *Proc. of ACM SIGCOMM*, 1998.
9. Rampal et al. Flow grouping for reducing reservation requirements for guaranteed delay service. *Internet Draft:draft-rampal-flow-delay-service-01.txt*, July 1997.
10. D. Ferrari, A. Benerjea, and H. Zhang. Network support for multimedia - a discussion of tenet approach. *Computer Networks and ISDN*, 26:1267–1280, 1994.
11. L. Georgiadis, R. Guerin, V. Peris, and K. Sivaranjan. Efficient Network QoS Provisioning Based on per Node Traffic Shaping. *IEEE/ACM Trans. Networking*, pages 482–501, August 1996.
12. S. Kweon and K. Shin. Providing Deterministic Delay Guarantees in ATM Networks. *IEEE/ACM Trans. Networking*, 6(6):838–850, December 1998.
13. A.K. Parekh and R.G. Gallager. A generalized processor sharing approach to flow control in integrated services networks: The single-node case. *IEEE/ACM Trans. Networking*, 1:344–357, June 1993.
14. D. Stiliadis and A. Varma. Rate-Proportional Services: A Design Methodology for Fair Queuing Algorithms. *IEEE/ACM Trans. Networking*, April 1998.
15. I. Stoica and H. Zhang. Providing Guaranteed Services Without Per Flow Management. *Proc. of ACM SIGCOMM*, Sept. 1999.
16. S. Vutukury and J.J. Garcia-Luna-Aceves. A Scalable Architecture for Providing Deterministic Guarantees. *Proc. of ICCCN*, Oct. 1999.
17. S. Vutukury and J.J. Garcia-Luna-Aceves. A Multipath Framework Architecture for Integrated Services. *Proc. IEEE GLOBECOM*, 2000.
18. S. Vutukury and J.J. Garcia-Luna-Aceves. A Traffic Engineering Approach to Minimum Delay Routing. *Proc. of ICCCN*, Oct. 2000.
19. L. Wang and et al. RSVP Refresh Overhead Reduction by State Compression. *Internet Draft, draft-wang-rsvp-state-compression-03.txt*, March 2000.
20. H. Zhang and D. Ferrari. Rate-Controlled Service Disciplines. *Journal of High Speed Networks*, Feb. 1994.
21. L. Zhang and et al. RSVP: A New Resource Reservation Protocol. *IEEE Communications Magazine*, 31(9):8–18, 1993.
22. Z. Zhang and et al. Decoupling QoS Control from Core Routers: A Novel Bandwidth Broker Architecture for Scalable Support of Guaranteed Services. *Proc. of ACM SIGCOMM*, pages 71–83, 2000.

Definition and Experimental Evaluation of an Architecture for Joint Quality of Service Control in Multimedia Networks

Franco Davoli, Daniele Luscardo, Piergiulio Maryni, and Angelo Pietra

Department of Communications, Computer and System Sciences
DIST-University of Genoa
Via Opera Pia 13, 16145 Genova, Italy
`(franco,luxi,pg,angelo)@dist.unige.it`

Abstract. We introduce a control architecture in which several (independent) multimedia clusters share the same (local or metropolitan) networking resources in a controlled framework. In particular, a central entity (i.e., the Gatekeeper) harmonizes the transmission rates of the various clusters following a given sharing policy. Each cluster, in turn, adopts its own end-to-end rate control mechanism to meet the Gatekeeper's transmission rate indications. A testbed has been developed and the system has been evaluated with real experiments by using different types of transmission sources. A software architecture is also introduced and described, with particular reference to the middleware framework realized with the Jini system.

1 Introduction

The near future in the telecommunications domain will be characterized by the coexistence of a large number of multimedia applications, all sharing parts of the same networking infrastructure [1]. To date, several efforts have been addressed in identifying end-to-end control algorithms able to optimize, in real time, transmission parameters such as transmission rate or coding settings (among others, see, e.g., [2,3], VIC [4], NEVOT [5], and NEVIT [6]). In the Wide Area Network (WAN) environment, DiffServ/IntServ [7,8,9,10] paradigms are being used to ensure Quality of Service (QoS) guarantees at the network level, but the use of end-to-end congestion control is still needed to avoid negative effects and the risk of collapse [11,12] caused by uncontrolled best-effort traffic. However, most of the work has been focused on controlling single multimedia sessions (i.e., the single transmitter-receiver(s) cluster), with the possible result of having a number of quality of service control algorithms running together in the same network in a non-cooperative way. Indeed, in limited frameworks such as Local Area Networks (LAN) or Metropolitan Area Networks (MAN), some form inter-operation would be possible, easy and helpful, with the aim of reducing congestion and providing also some level of QoS in an otherwise uncontrolled segment of the network. In this scenario, which is roughly sketched in Figure 1, we introduce a control architecture in which several (independent) multimedia clusters share the same networking resources in a controlled framework. In particular, a central entity

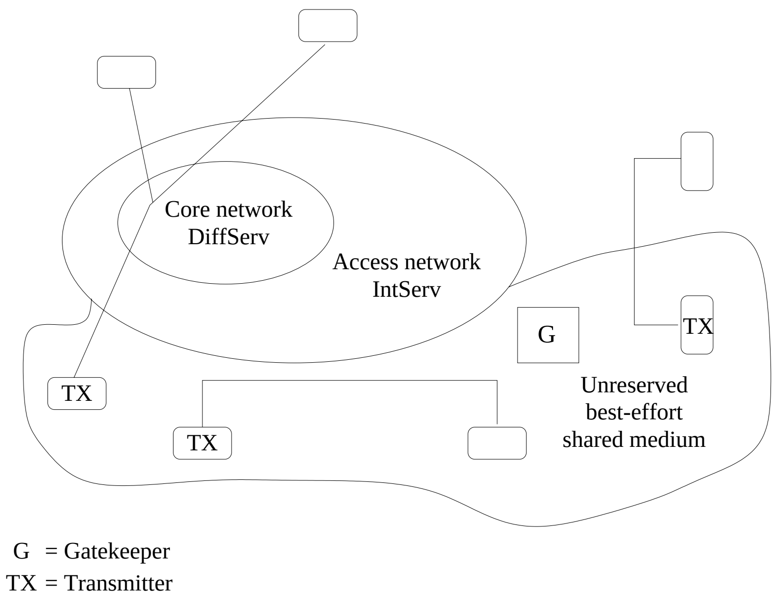


Fig. 1. The system scenario.

(i.e., the Gatekeeper [13]) harmonizes the (relative) transmission rates of the various clusters following a given sharing policy. Each cluster adopts its own end-to-end rate control mechanism to meet the Gatekeeper's transmission rate indications. A large number of end-to-end control mechanisms have been proven to fit in this architecture. A testbed has been developed and evaluated with real experiments over a 100 Mbit/s LAN. The results presented demonstrate the effectiveness and the flexibility of the system. The software architecture is outlined focusing, in particular, on the Jini middleware framework, which has been adopted for the implementation of the testbed.

The paper is organized as follows. Section 2 describes the system control architecture, also outlining a number of admission control issues. Section 3 illustrates the software architecture used for the implementation of the testbed. It focuses, in particular, on the middleware issues. Section 4 shows the results obtained from the testbed system, highlighting the differences between controlled and uncontrolled systems (with respect the control carried out by the Gatekeeper).

2 The Control System Architecture

The system under control is characterized by a set of multimedia clusters composed by a main transmitter and one or more receivers (in a multicast fashion). The multimedia transmission within the system is ruled by a two level rate control scheme. The first level of rate control is embedded within each single multimedia cluster and takes place as a result of an exchange of control messages between the transmitter and the receiver(s) (see also, e.g., [14]). The implementation details of such control algorithm are not of interest for the overall control system: each cluster can decide its own rate control procedure depending on the particular type of multimedia session on service. The second level of rate control is performed by a network entity, which is embedded into the Gatekeeper, that supervises the activity of all controlled multimedia clusters, as shown in Figure 2. Each cluster (i.e., the transmitter), at the beginning of the session, registers with

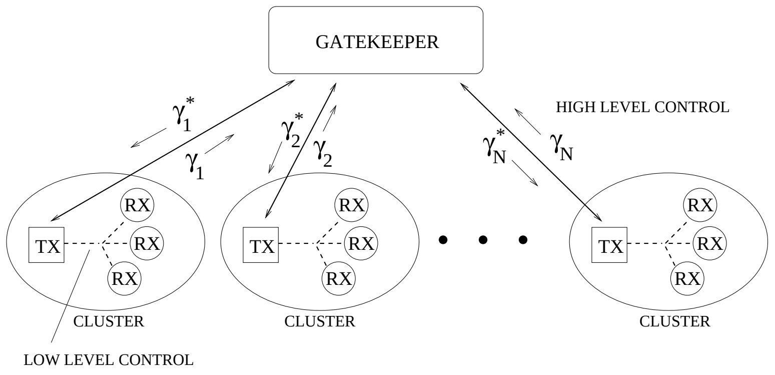


Fig. 2. The system control model.

the Gatekeeper, declaring its transmission rate boundaries, say $r^{\min}$ and $r^{\max}$. For the i-th cluster (out of a total number of N controlled ones), the Gatekeeper periodically computes a control parameter γ_i^*, $i = 1, \ldots, N$, which indicates the amount of bandwidth r_i^* the transmitter should conform to. More in particular:

$$r_i^* = \gamma_i^* \cdot r_i^{\max} \qquad\qquad 0 \le \gamma_i^* \le 1 \ . \tag{1}$$

The control parameter that the Gatekeeper computes is, of course, just a reference target. Each transmitter tries to set its transmission rate accordingly, but it is not certain it will succeed. The only real constraint is that such reference target should not be exceeded. Furthermore, each transmitter might decide to decrease its transmission rate depending on network conditions. For this reason, the Gatekeeper polls the transmitters in order to receive a feedback value γ_i ($0 \le \gamma_i \le 1$), which is a number such that $r_i = \gamma_i \cdot r_i^{\max}$. The rate r_i is the actual rate for cluster i.

By means of its control parameters $\underline{\gamma}^* = [\gamma_1^*, \gamma_2^*, \ldots, \gamma_N^*]$ the Gatekeeper has the ability to implement a large number of policies. Just as an example, a very simple computing scheme for the values of $\underline{\gamma}^*$ could be the following (which gives rise to the same control parameter for all clusters):

$$\gamma_i^* = \frac{\sum_1^N \gamma_j r_j^{\max}}{\sum_1^N r_j^{\max}} \qquad i = 1, \ldots, N \ . \tag{2}$$

Each transmitter i then scales back the value of γ_i^* to its transmission range by means of (1). In other words, though all γ_i^* are equal in this simple case, the actual rates are generally different; only the scaling factor is unique.

So far so good. The various clusters operate independently, with the only constraint that the reference rate fed by the Gatekeeper should not be exceeded. However, it is reasonable to expect that, due to network bandwidth oscillations, one or more clusters might decrease their transmitting rate. If a mechanism such as the one in (2) is used, then the reference value computed by the Gatekeeper is forced to decrease (since the values of $\gamma_i, \ i = 1, \ldots, N$ decrease). This fact is positive because the system reallocates bandwidth among the clusters following the desired policy. However, when bandwidth becomes again available, some sort of mechanism is needed to raise back the cluster rates. Among the many possible ones, the mechanism used here is the following. If the feedback values $\gamma_i, \ i = 1, \ldots, N$ remain constant for more than K times (i.e., K polling events), then the Gatekeeper increases its computed control parameters by β (provided that the parameter remains less than one). In our experiments, for the sake of simplicity, the increasing value β has been set equal for all clusters. This mechanism has proven to work pretty well, as shown in Section 4. A further issue to be addressed is the possible presence of small oscillations. This might happen for the effects of (2) if station j does not reach exactly its value γ_j^*: the other clusters will be forced to decrease their rate. A trivial solution is to allow transmitters to get just *close* to their target. To do so, it is sufficient to substitute, in (2), γ_j with $\hat{\gamma}_j$ where

$$\hat{\gamma}_j = \begin{cases} \gamma_j^* \text{ if } \ \gamma_j^* - \gamma_j < \eta \\ \gamma_j \text{ otherwise} \end{cases} \ . \tag{3}$$

and η represents an "activation" threshold.

A form of high level control activity such as the one operated by the Gatekeeper is particularly important when the available bandwidth in the network changes abruptly and the control algorithms of the various clusters have to operate in order to achieve again transmission stability. A key example is the joining of a new cluster. If all clusters operated in a separate way, their stability points would be rather undefined and uncontrollable. Depending on their particular control algorithms, the stability levels of clusters that would have needed almost the same bandwidth requirements might be remarkably different. The Gatekeeper helps in harmonizing them by introducing a simple form of global coordination.

A nice characteristic of this control scheme is that the interface between a cluster controller and the Gatekeeper is only represented by a positive fraction number, less than one. A better knowledge of the low level control algorithms would, of course, lead to a finer high level cluster handling, but it would also make the interface more complex. The degree of freedom in the choice of the cluster control algorithm would be also much lower. A drawback of this scheme is that low level control algorithms have to be designed to try to conform to the Gatekeeper control parameter. This problem can, however, be easily solved by introducing some sort of "middle-layer" control level which "translates" this single parameter value into an appropriate set of controls for the particular transmitting control process (see, e.g., [14]). Another drawback is that such a single parameter might, in effect, not allow to tune the cluster with a high degree of accuracy. Nonetheless, as will be shown in the remainder of the paper, the results obtained by applying this kind of architecture are definitely encouraging.

2.1 Admission Control Issues

The Gatekeeper also performs the admission control function for new clusters that ask to participate in the overall control system. The admission control for multimedia sessions is one of the most discussed issues in the literature (see, e.g., [15]). The main problem is the lack of valuable models and, hence, the need to eventually come up with heuristics or measurement-based techniques that may not always give high quality results, if not properly engineered. Even if this work is not focused on admission control issues, we describe in the following how this function has been performed in our model.

For the system being described in this work, the model that best fit is the "classic" one introduced in [16]. In this case, in particular, the admission controller relies on the concept of (Multimedia) Schedulable Region, which is a data abstraction that describes the maximum load achievable by a device or a network. The model needs the identification of a fixed and limited number of traffic classes. A traffic class gathers streams with similar characteristics in terms of bandwidth requirement, traffic behavior, and quality of service constraints. For example, the set of MPEG streams might identify a traffic class. The identification of streams that have similar requirements is receiving more attention to date, also in view of the way real-time traffic is going to be handled in the Internet (e.g., via the Multiprotocol Label Switching [17]). The Multimedia Schedulable Region describes the region, in the space of the traffic classes, within which packet-level quality of service constraints are guaranteed. An example of Multimedia Schedulable Region for three classes of traffic is given in Figure 3 below. The nice characteristic of this model is that it allows to maintain a neat separation between the admission controller and the underlying network details. In the simplest form, the admission controller, upon a request for a new multimedia session, merely checks if the new state would be inside the region in order to decide for acceptance or rejection. However, more sophisticated forms are possible (see, e.g., [16]).

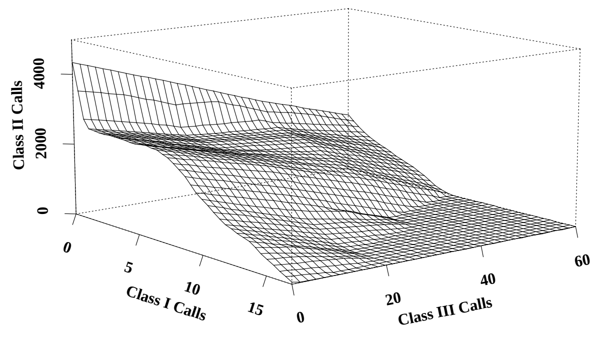


Fig. 3. An example of Multimedia Schedulable Region for three traffic classes.

The problem now turns into the identification (or estimation) of the boundaries of the Multimedia Schedulable Region, i.e., the identification of the maximum load bearable by the network. Several solutions have been described in the literature (see, e.g., [18,19,16]). Some of them make use of analytical procedures (such as [19]), while others rely on heuristic models (see, e.g., [20]) or measurement based techniques (as in [21,22]). Our experience has suggested that the latter ones have better performances, especially when results are needed for real time control purposes. The Multimedia Schedulable Region would work perfectly in a completely controlled system, where no background transmission "noise" is present. The region would remain fixed over time. In this context, where we allow some form of (light) background traffic, the need is to keep the region updated. It might enlarge if the background traffic decreases, or it might decrease if the background traffic increases. The method we adopted here is based, in particular, on measurements taken over a test flow. An interesting discussion that highlights peculiarities and advantages of this well-known technique can be found, e.g., in [23] and [24]. The test flow is used in this context for indicating if a given shape of the Multimedia Schedulable Region still holds. For a detailed description of the test flow measurements technique, see [24]. For the sake of completeness, however, a short presentation of it is reported in the Appendix.

The main result obtained by the test flow is a value $\xi_\epsilon(t)$ that increases as the total load in the network increases. It is possible to find, for each traffic class, a threshold value above which no other sessions of that class can be accepted. If the state of the Multimedia Schedulable Region would indicate the availability of space for a given session but $\xi_\epsilon(t)$ has exceeded (or is close) to such threshold value, the Multimedia Schedulable Region is decreased. If, on the other hand, given a previously reduced Multimedia Schedulable Region, the value of $\xi_\epsilon(t)$ is below another threshold, then the region is augmented (till it reaches the "original maximum size").

3 Implementing the Architecture

The overall architecture involves a number of actors that have to co-operate together within the control system. The main ones are the Gatekeeper and the cluster transmitters, but also the test flow handling processes participate, and the implementation has to leave room for other actors that might be added in the future. It is evident, hence, that all of them have to operate above a middleware layer that takes care of binding requests with responses throughout the network. Strictly speaking, the core of a middleware service is a lookup server that accepts registrations from whatever entity that is able to provide services through the network (from printers to, e.g., computation services), and binds such services, on request, to other entities that require it. A key point of a middleware architecture is, hence, the protocol that allows the interactions with such lookup service and that defines the way the interfaces of the various service providers have to be passed around, through the network. The lookup service, moreover, has to have a way to be known within the network.

The testbed we have developed for validating the aforementioned model architecture has been implemented using the Java framework. More in particular, the key packages heavily employed have been the Java Media Framework (JMF) [25] and the middleware Java environment Jini [26]. The latter represents a complete setting (at least, for our requirements) for the implementation of a distributed service system coordinated by a lookup service. There are three key moments in which Jini intervenes. The first is the registration of the various services. Each actor, at start-up, registers by the lookup service as shown in Figure 4. Notice that each software component is unaware to act within a Jini environment: the middleware system is hidden by a tiny software interface layer, as shown in the figure. Figure 4 gives also the opportunity to start describing the transmitter's main components. They are structured by following the JMF architecture as a sending engine, that takes care of handling the particular input device and that shows itself as a generic Java datasource. The session manager is the one that takes care of Quality of Service handling in co-ordination with the peer session manager(s) at the receiver site(s), if any. The Real Time Protocol (RTP) and Real Time Control Protocol (RTCP) are originated within the session manager, which has also the ability to drive the sending engine. The Jini Tx Interface is the tiny software interface layer that makes transparent the presence of the Jini middleware framework (the same is for the other Jini interface modules). Once actors have registered their services (the Jini interfaces do this), they are ready to operate in the network. The set-up of a new cluster session requires the main transmitter to register by the Gatekeeper (notice that this has nothing to do with middleware: it is the admission control function). In order to do this, the transmitter looks for the Gatekeeper registry interface in the lookup service, it downloads it, and it uses it to ask the Gatekeeper for admission. The Gatekeeper has previously asked for the interface to interact with the test flow module, as schematically shown in Figure 5. Now a cluster session can take place. The transmitter asks for the interface that allows to receive the control parameter by and to send feedback to the Gatekeeper (so does the Gatekeeper

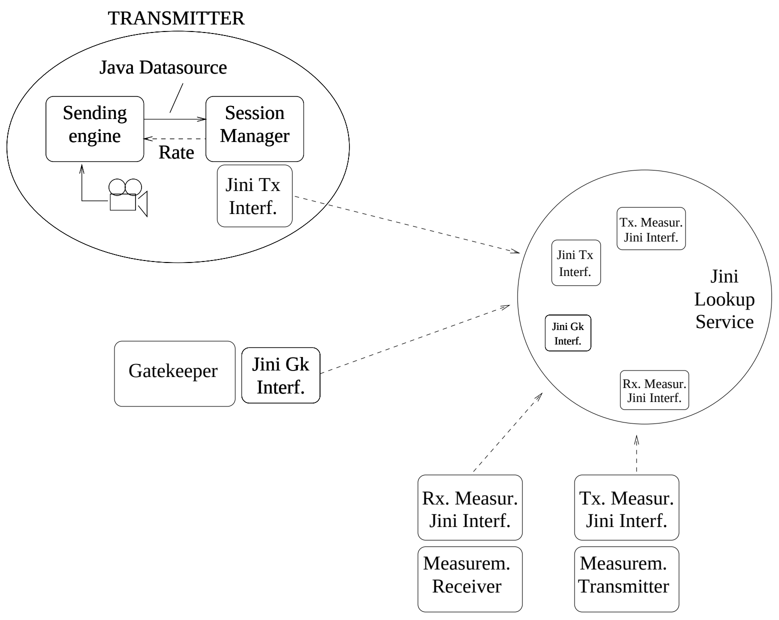


Fig. 4. The (initial) registration phase within the control architecture.

with respect to the transmitter). It then sets up a real time connection with a requesting receiver, which also has found the way to connect to the transmitter in the lookup service server. This step is sketched in Figure 6.

4 Numerical Results

The system has been tested within a 100 Mbit/s switched Ethernet. The load was composed by uncontrolled background traffic (generated by other users), by controlled real-time multimedia sessions, and, sometimes, by dummy *ftp* sessions. Two types of video streams have been mainly considered: the first one is a low bandwidth H.263 [13] video stream with maximum transmission rate of 100 Kbits/s, while the second is a medium bandwidth MJPEG [27] one, with maximum transmission rate of 1 Mbits/s. Audio streams, in MP3 format [13], are also currently under investigation. Each cluster was sending a single video stream and, hence, by having a mix of clusters running in parallel, we also had a mix of H.263 and MJPEG in the network. In the experiments, clusters were coming up asynchronously in order to see both the disturbing effect and the ability to reach a stable transmission rate.

The end-to-end control algorithm implemented in the transmitters is the one adopted for the VIC [4] application, as introduced in [28], with the modification that the source also accepts a control value by the Gatekeeper. Real-time streams are transported over a RTP/UDP stack and the source, on receiving an RTCP

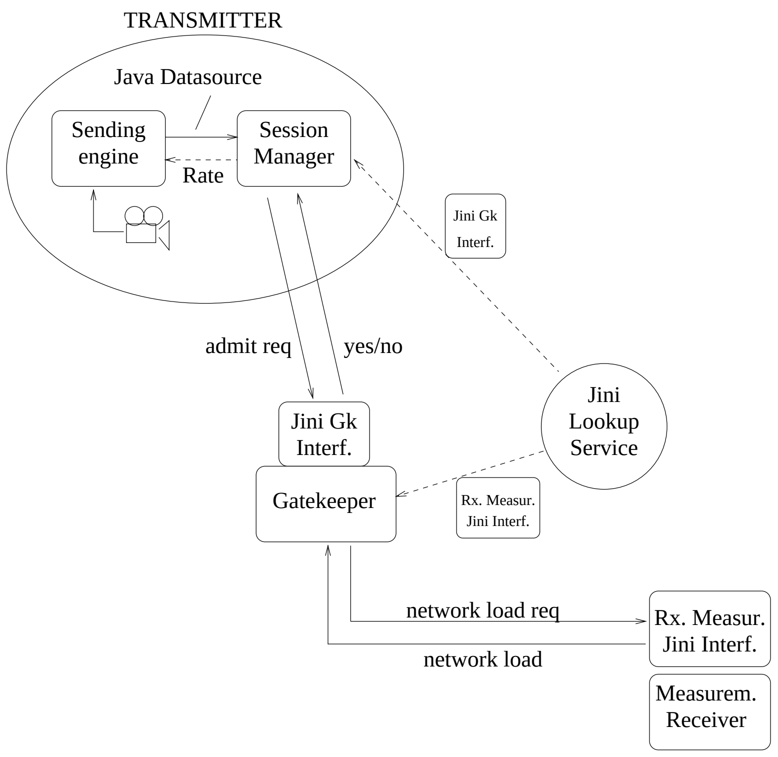


Fig. 5. The admission control function.

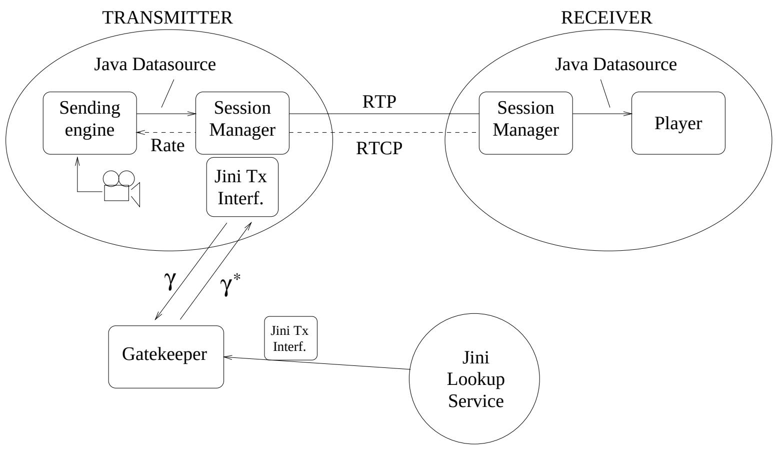


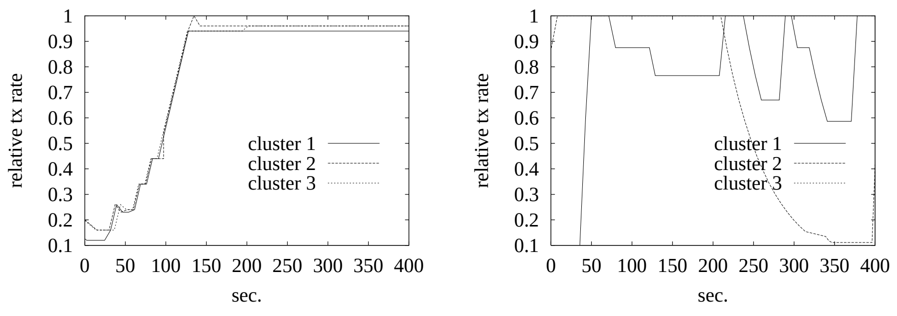
Fig. 6. The multimedia cluster session.

report, performs an RTCP analysis, estimates the network state, and adjusts the bandwidth (coherently with the threshold given by the Gatekeeper). All steps except the adjustment are independent of the characteristics of the multimedia application. The adjustment is, of course, dependent, in that a controller has to drive the bit error generation of the coder, and this is done in different ways for different stream types. For example, the H.263 coder allows a direct control of the bit rate; the MJPEG coder we have used is driven by a parameter that, in our case, has an almost linear correspondence with the bit rate. In general, this control framework applies smoothly whenever the source bit rate can be directly selected. Recent multimedia coding schemes almost always allow to do so.

The Gatekeeper performs the high-level control scheme with the activation threshold η (see, equation (3)) set to 0.05. The Gatekeeper recomputes and redistributes the control value roughly about every 7 sec, which is the standard time RTCP messages arrive to the source from the player(s). Whenever the value of the control parameter is met by the clusters for $K = 2$ times (i.e., every 14 sec), then the control parameter is augmented by a value $\beta = 0.1$.

In general, the same experiment has been repeated many times, both in the controlled and in the uncontrolled situation, and the difference between the two cases was always evident.

Figure 7 shows the results obtained from three clusters that use H.263 video streams, both in the controlled and uncontrolled case. The load in the network is enforced with a number of *ftp* sessions. The usefulness of the joint high level control performed by the Gatekeeper is evident. Without it, the various clusters behave in a completely independent fashion, leading to a highly unfair resource usage.

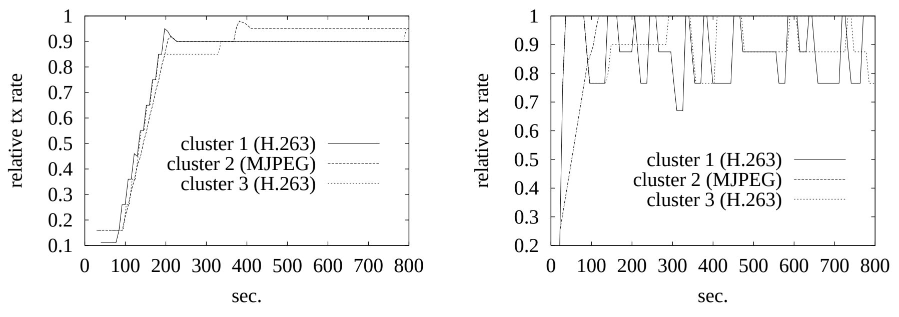


(a) controlled (b) uncontrolled

Fig. 7. Controlled vs. uncontrolled system. All clusters send H.263 video streams

In Figure 8 a mix of different types of sources is presented. In particular, two of them transmit H.263 streams, while the remaining one transmits MJPEG.

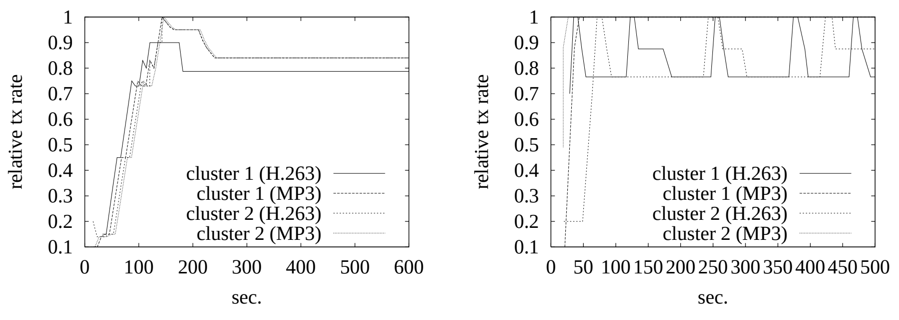
The results are comparable with the ones presented in Figure 7. Notice that, due to the relative transmission parameter, streams that have different bandwidth requirements can be treated together.



(a) controlled (b) uncontrolled

Fig. 8. Controlled vs. uncontrolled system.

Figure 9 shows results obtained by having clusters composed of two streams, namely, an H.263 video stream and a high quality MP3 audio one. The maximum rate of the MP3 stream is 128 Kbits/s. The plots, again, show the benefits of the presence of the high level joint control mechanism.



(a) controlled (b) uncontrolled

Fig. 9. Controlled vs. uncontrolled system, with audio streams.

The policy carried out by the Gatekeeper has to be studied carefully. In fact, there are possible negative effects that may arise from a very simple implementation such as the one considered so far. One of the main drawbacks can originate by the fact that the current policy tends to move all clusters' flows together and, hence, if one cluster decreases its rate for reasons that are independent of congestion, the Gatekeeper would force also other clusters to follow. This effect is caused by the coupled action of the weighted averaging mechanism used to derive the control parameter and of the feedback from the clusters. Such behavior is depicted in Figure 10, where the MJPEG source (the "heaviest" in terms of load) decreases its rate even if the available bandwidth of the network remains high (this could happen, for example, for coder failures). A simple load estimation mechanism, such as the one introduced in the Appendix, could help in distinguish whether a source decreases its rate because of congestion or not.

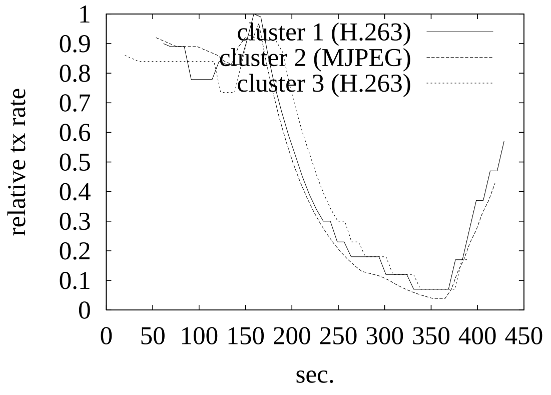


Fig. 10. The effect of a transmitter failure

5 Conclusions

An overall control architecture for the joint control of multimedia clusters has been introduced. The overall control actions take place at two different levels: the intra-cluster and the inter-cluster. At the intra-cluster level, the control mechanism takes care of adapting the transmission rate based on end-to-end traffic considerations. At the inter-cluster level, a central actor, embedded in the Gatekeeper, drives all the cluster by following a given policy, in order to reduce congestion and to improve QoS for the multimedia flows in a best-effort environment. A possible software implementation is also introduced. The presented results show the effectiveness of a central, high-level, control action, along with the possible negative effects of too simplistic high-level control policies.

6 Acknowledgment

This work was supported by the Italian Ministry of University and Scientific and Technological Research (MURST), under the National Research Program on Techniques for QoS Guarantees in Multiservice Telecommunication Networks.

7 Appendix: The Test Flow Measurement Technique

Consider two end-points of a "channel" of a generic network, one of them acting as a *transmitter*, the other one acting as a *receiver*. We may think of a "channel" as a link between any two nodes in a network (which may be shared with other stations, as in a LAN) or a path composed by more than one link (which we will consider as a "virtual pipe"), which a certain total bandwidth can be attributed to. The transmitter generates and sends a short packet to the receiver once every T seconds. The receiver, upon reception of a packet, computes its interarrival time, which will be used to construct an estimate of the load of the channel. In such a framework, if the channel is lightly loaded, it is reasonable to expect that also the receiver will "see" a synchronous (or quasi-synchronous) train of packets. On the other hand, as the load of the channel increases, we expect to see, at the receiver site, a degradation of such a synchronicity. A detailed description of how the received packet train can be analyzed is given in [24]. We only sum up here the main outcomes. The major observation is that the process of the interarrival times presents self-similar characteristics and, as such, considering first or second moment averages (over windows of samples) leads to poor results. A much better value is the ϵ-percentile $\xi_{\epsilon,n}(i)$, which is the i-th percentile of n sliding variances computed over non overlapping windows of m interarrival times. The value $\xi_{\epsilon,n}(i)$ is such that 100ϵ of such variances are less than $\xi_{\epsilon,n}(i)$. This value is easy to compute, it gives a stable load estimate and it is also quite fast in reacting to varying load conditions. Figure 11 shows an example in which the network is incrementally loaded with three heavy file transfer sessions.

It is worth observing that the synchronous packet train can be thought as the equivalent of a "fuse" in an electric implant. It is very sensible to load conditions and, indeed, we have noticed it is the first one to "break" in case of congestion (even when other real-time applications keep working well). It could be, hence, used to some extent as an "alarm" flow.

Self-similar parameters such as the Hurst one can be also evaluated and, indeed, they reflect the system load. However, when it comes to exploit self-similar analysis, we have to remember that many time scales are involved. Hence, they are scarcely applicable in a real-time framework, where the estimating mechanism must be fastly reactive to load variations. Nonetheless, they might be useful in estimating some level of "background" traffic, over which multimedia connections may be thought of as being "superimposed". Their application in this direction is currently under study.

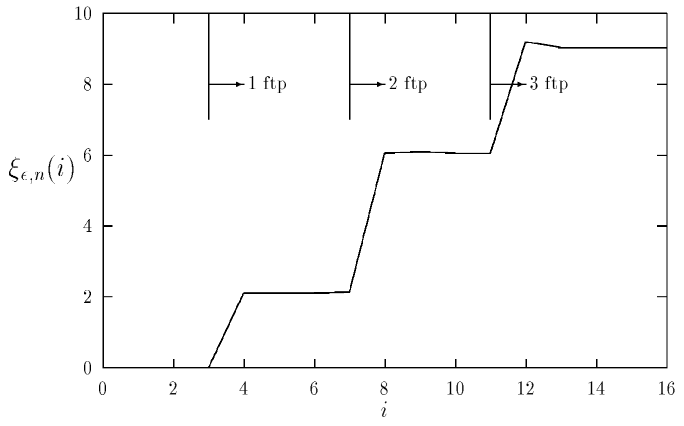


Fig. 11. Plot of $\xi_{\epsilon,n}(i)$, $m = 10$, $n = 30$, $\epsilon = 0.3$.

References

1. M. Decina and V. Trecordi. Convergence of Telecommunications and Computing to Networking Models for Integrated Services and Applications. *Proceedings of the IEEE*, pages 1887–1914, December 1997.
2. D. Sisalem. End-To-End Quality of Service Control Using Adaptive Applications. In *Proc. Fifth Int. Workshop on Quality of Service (IWQOS-97)*, New York, May 1997.
3. D. Sisalem and H. Shulzrinne. The Loss-Delay Based Adjustmkent Algorithm: A TCP-Friendly Adaptation Scheme. In *Proc. International Workshop on Network and Operating System Support for Digital Audio and Video (NOSSDAV)*, pages 215–226, Cambridge, July 1998.
4. S. McCanne and V. Jacobson. Vic: A Flexible Framework for Packet Video. In *Proc. ACM Multimedia-95*, Nov 1995.
5. S. Baker. Multicasting for Sound and Video. In *Unix Review*, volume (12):2, February 1994.
6. D. Sisalem, H. Shulzrinne, and C. Sieckmeyer. The Network Video Terminal. In *Proc. Fifth IEEE International Symposium on High Performance Distributed Computing*, Syracuse, N.Y., August 1996. IEEE Computer Society.
7. S. Blake, D. Black, M. Carlson, E. Davies, Z. Wang, and W. Weiss. An architecture for differentiated services. *Internet Engineering Task Force, Request for Comments, RFC 2475*, Dec. 1998. [Online]. Available FTP: ds.internic.net/rfc/rfc2475.txt.
8. T. Li, Y Rekhter. A provider architecture for differentiated services and traffic engineering (PASTE). *Internet Engineering Task Force, Request for Comments, RFC 2430*, Oct. 1998. [Online]. Available FTP: ds.internic.net/rfc/rfc2430.txt.
9. P.P. White, J. Crowcroft. The Integrated Services in the Internet: state of the art. *Proc IEEE*, 85(12):1934–1946, Dec. 1997.
10. R. Braden, D. Clark, S. Shenker. Integrated Services in the Internet architecture: an overview. *Internet Engineering Task Force, Request for Comments, RFC 1633*, June 1994. [Online]. Available FTP: ds.internic.net/rfc/rfc1633.txt.
11. S. Floyd and K. Fall. Promoting the Use of End-to-End Congestion Control in the Internet. *IEEE/ACM Transactions on Networking*, 7(4):458–472, August 1999.

12. S. Floyd. Congestion Control Principles. Technical report, IETF - INTERNET DRAFT, June 2000.
13. C. Perey. *Videoconferencing Standard (Digital Multimedia Standards Series)*. Chapman & Hall, April 2001. ISBN: 0412148412.
14. A. Iscra and S. Zappatore. Experimental Performance Evaluation of a Flow Control Algorithm for Multimedia Network Applications over a WAN. In *Proc. of Symp. on Perf. Eval. of Computer and Telecomm. Syst. (SPECTS 2000)*, Vancouver, CA, June 2000.
15. N. Shacham and H. Yokota. Admission Control Algorithms for Multicast Sessions with Multiple Streams. *IEEE Journal on Selected Areas in Communications*, 15(3):557–566, April 1997.
16. J.M. Hyman, A.A. Lazar, and G.Pacifici. A separation principle between scheduling and admission control for broadband switching. *IEEE Journal on Selected Areas in Communications*, 11(4):605–616, May 1993.
17. Special Issue. Multiprotocol Label Switching. *IEEE Communications Magazine*, 1999.
18. D. Ferrari and D. C. Verma. A scheme for real-time channel establishment in wide-area networks. *IEEE Journal on Selected Areas in Communications*, SAC-8(3):368–379, April 1990.
19. R. Guerin, H. Ahmadi, and M. Naghshineh. Equivalent capacity and its application to bandwidth allocation in high-speed networks. *IEEE Journal on Selected Areas in Communications*, 9(7):968–991, September 1991.
20. A. Parekh and R. Gallager. A generalized processor sharing approach to flow control in intraged services networks - the single node case. In *Proceedings of the IEEE INFOCOM*, pages 915–924, Florence, Italy, May 1992.
21. S. Jamin, P.B. Danzig, S. Shenker, and L. Zhang. A Measurement-Based Admission Control Algorithm for Integrated Service Packet Networks. *IEEE/ACM Transactions on Networking*, 1997.
22. P. Maryni and G. Pacifici. Real-Time Estimation of the Link Capacity in Multimedia Networks. In *Proceedings of the IFIP 5th International Workshop of Quality of Service (IWQOS'97)*, New York, May 1997.
23. V. Paxson. On calibrating measurements of packet transit times. In *SIGMETRICS '98 / PERFORMANCE '98 Joint International Conference on Measurement and Modeling of Computer Systems.*, volume 26/1, pages 11–21, Madison, WI, USA, June 1998. ACM.
24. P. Maryni and F. Davoli. Load Estimation and Control in Best-Effort Network Domains. *Journal of Network and Systems Management*, 2000. (to appear on Dec. 2000).
25. R. Gordon, S. Talley, and R. Gordon. *Essential JMF - Java Media Framework*. Prentice Hall, 1998.
26. W.K. Edwards. *Core Jini*. The Sun Microsystems Press Java Series. Prentice Hall, Upper Saddle River, 1999. ISBN: 0-13-014469-X.
27. ITU-T. *M-JPEG in future Networked Television Production*, 1998. [11B/13, 10-11R/16] EBU Statement D82-1998.
28. I. Busse, B. Deffner, and H. Shulzrinne. Dynamic QoS Control of Multimedia Applications based on RTP. In *Proc. of First Int. Workshop on High Speed Networks and Open Distributed Platforms*, S. Petersburg, Russia, 1995.

Quality-of-Service Guarantees for Multicast Traffic in Heterogeneous Multi-service Networks*

Andrea Borella[1], Giovanni Cancellieri[1], Elena Pagani[2], and Gian Paolo Rossi[2]

[1] Dip. di Elettronica e Automatica, Università degli Studi di Ancona
cangio@popcsi.unian.it

[2] Dip. di Scienze dell'Informazione, Università degli Studi di Milano
{pagani, rossi}@dsi.unimi.it

Abstract Multimedia and real-time applications have peculiar requirements in terms of the quality of data transmission services, that are not satisfied by the best effort nature of IP. Moreover, many of the multimedia applications are characterized by a multicast communication pattern. In this work, we propose a functional architecture aiming at providing a common framework to deploy network protocols supporting QoS. The architecture is compliant with the main standards proposed in the literature. We discuss two possible implementations of the architecture modules for the service set-up, in either the int-serv model or the diff-serv model. We present some performance evaluations obtained by performing experiments with those two implementations.

1 Introduction

Multimedia applications are becoming of common use in the last few years, thanks to the large diffusion of the World Wide Web. Although the possibility of accessing those applications is attracting a lot of interest by the potential users, so far the tools to provide them with a good quality level are not yet available. Despite the continuous growth of the link capacity and of the switching power of the routers, the best effort nature of IP is not sufficient, alone, to meet the requirements (e.g., in terms of received rate, maximum end-to-end delay or maximum data loss probability) of the multimedia and the real-time applications.

As a further problem, many of the multimedia applications are characterized by either a one-to-many or a many-to-many communication pattern. Hence, they strongly demand for *multicast* protocols, that allow both to ease the management of several recipients and to efficiently use the network resources. Multicast introduces new quality parameters (e.g., the *fair delay*) to be considered when providing QoS, and new issues in the QoS protocols design, such as that of having to deal with heterogeneous recipients.

* This work was supported by the MURST under Contract no.9809321920 "Transport of multicast packets with QoS guarantees".

Several initiatives have been undertaken in recent years to bridge this *Quality Of Service* (QoS) gap [16,15,8,6,3]. So far, however, the standardization process is far to be completed, the available proposals do not refer to a common architectural framework and the literature is fragmented. Moreover, the available protocols apply to single, homogeneous, QoS-domains, while the understanding of the inter-domain communications is at a preliminary stage. All these arguments are slowing the deployment of large QoS platforms by ISPs and making very hard to ensure their interoperability.

In this paper, we describe a functional and architectural framework for end-to-end QoS in large multi-domain networks where int-serv and diff-serv are both deployed, or deployable, to provide service differentiation. The architecture provides the building blocks to design the structure of the routers operating at the border between domains adopting different differentiation policies. It guarantees backward compatibility with existing protocols and services. We studied possible mechanisms to realize the fundamental modules of the architecture for the service set-up under both the IS and the DS models. The performance of those mechanisms has been evaluated by simulations.

The paper is organized as follows: in Section 2, we introduce the model of the system we consider and the notation used throughout the paper. In Section 3, we discuss the reference functional architecture. In Section 4, we describe the architecture implementation both for the IS model and for the DS model. In Section 5, we present our experimental results.

2 The System Model

So far, two main service differentiation models have been provided: *integrated services* (int-serv) [6] and *differentiated services* (diff-serv) [3]. Int-serv's offer QoS on a per-flow basis. The data packets are labeled with the identifier of the flow to which they belong. By contrast, diff-serv offers QoS service on a per-packet basis. According to this model, each DS domain offers a specified set of service classes, and the traffic entering the domain is assigned to the offered service classes as the consequence of a classification and, possibly, conditioning process. Each packet, through the use of the TOS/*DS* byte field of the IP header [11], is labeled with the identifier of the class (*DS codepoint*) to which it belongs. Resource reservation is performed on the base of the different service classes. All the packets which belong to a given service class C are subject to the same forwarding policy; they constitute a *behaviour aggregate*.

The interconnected system we will consider in the following is composed of a set of networks connected by routers. The hosts in the networks may run applications that require a QoS transmission service. Routers allow the forwarding of messages through different networks and are assumed to implement the QoS services. In the following, the term QoS-*capable* routers is used to denote these QoS-sensitive routers. We call *domain*, a set of contiguous networks which operate with a common set of service differentiation models, provisioning policy and service definition. We denote as *border* routers those routers connecting contiguous network domains. In order to guarantee the requested end-to-end QoS, the *border* routers are responsible of mapping the service provisioning and forwarding policy to which a flow is subject in one domain into the model and policy which have to be applied in the adjacent domain. The translation policy depends on the agreement between the adjacent domains.

3 The Reference Architecture

In [13] we propose a reference functional architecture to support QoS in the Internet. It has been designed for large networks, where both IS and DS can be de-

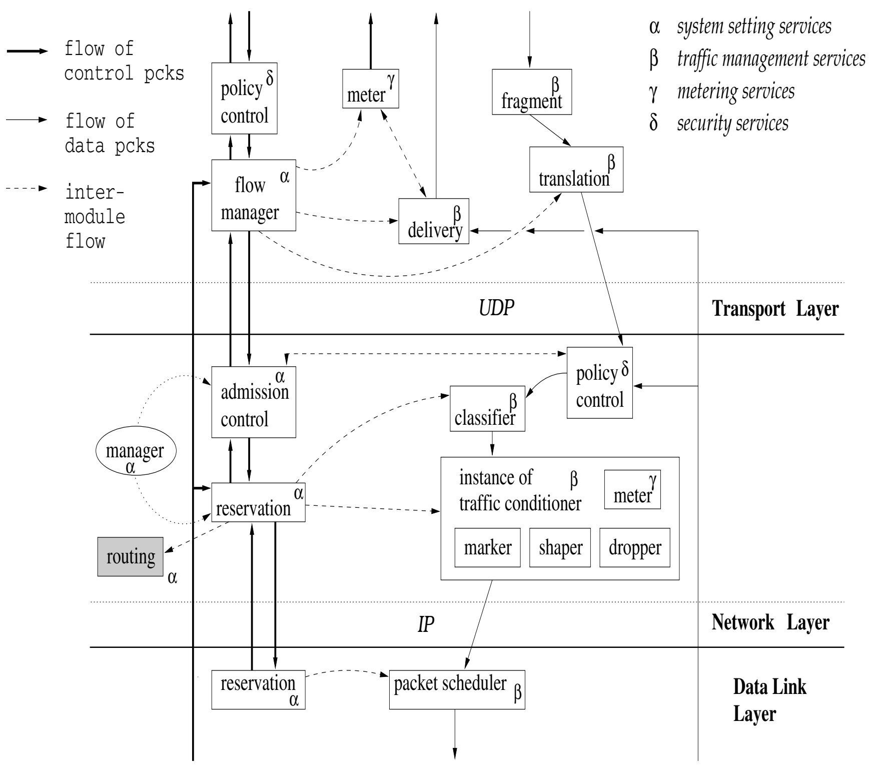


Fig.1. Functional architecture of a QoS-capable router

ployed. It supports both unicast and multicast communications. It is composed by the functional modules needed to provide QoS over an IP-based network. Most of the functional modules composing the described architecture are covered by the unicast-oriented and multicast-oriented protocols which have been proposed so far [15,16,8,10]. Other modules are available within either research or proprietary QoS architectures (e.g., MBone, XRM, the Tenet architecture, the Heidelberg QoS model). The modules organization corresponds to a layered structure which can be easily mapped onto the layers 2, 3 and higher, end-to-end, of the OSI reference model.

The overall functional organization of a QoS-capable element of a network is described by the QoS architecture of Figure 1. This architecture has been derived after the scanning of the literature and its functional re-organization aimed to be as general as possible, to guarantee backward compatibility with the available proposals, and to reduce (not eliminate yet) function duplications.

In Figure 1, bold lines are used to specify the control flow, which describes the *service set-up phase*, while thin lines specify the data flow, which describes the *data transfer phase*. Dashed lines indicate intra-layer control/data flows.

The functional modules that belong to this QoS architecture can be grouped in four main classes, spanning the layers of Figure 1.

System setting class. This class embodies the functions needed to reserve the buffer and bandwidth resources and to set the internal QoS parameters along the path from the source to the destination(s) of a QoS flow. Since applications are likely to be unaware of the network internals, at the end-to-end layer a *lieson* service should be provided between host and network functions. It consists in supplying an abstraction of the underlying QoS implementation. It translates the independent application service specification into the format which is suitable for the network layer (e.g., a `TSpec` and a `RSpec` [6] in the case of int-serv's, or a Per-Hop-Behavior (PHB) [3] in the case of diff-serv's). Resource allocation must be performed at all of the architectural levels, as well as the configuration of both the modules carrying out the packet processing (e.g. packet classification and scheduling) at each intermediate router, and the destination entity that delivers the data flow at the final user.

Traffic management class. This class groups the functions which define the QoS traffic management policy. The modules belonging to this class apply the policies for packet processing decided in the service set-up phase. They also deal with the tasks of message fragmentation and reassembly, data encoding and data delivery at the destination according to the agreed QoS.

Metering class. This class includes the functions which allow to monitor QoS parameters during both the set-up and the data transfer phases in order to trace the dynamics of resource availability to allow reservation, adaptation and exception handling. The metering services have multiple purposes. The gathered measures are used to estimate the amount of resources needed to achieve a given QoS. These estimates are used when a request for a new QoS flow or behaviour aggregate is received, to perform the reservation and the module configuration. The measurements may be as well used at both the end-to-end layer and the network layer to dynamically re-configure some of the functional modules to adapt to the changing network conditions. The reports might be notified to the source application, to allow traffic profile adaptation according to the available network resources.

Security class. This class includes functions which allow the authentication of an application, access control, accounting and security management. At the network layer, the validity of all the incoming requests (from both the network and the upper layer) is checked. Whenever a new data packet is received, it must be checked the correctness of a specified portion of the packet header and its coherence with the QoS requested at the set-up time.

For more details about the functional architecture, the interested readers may refer to [13]. In that work we show as well how the proposed architecture can be applied to provide interoperability among different domains.

3.1 Supporting QoS-Multicast Traffic

In Figure 2, we show how the network layer modules of a generic QoS capable router should be modified with respect to Figure 1 to support multicast traffic. Modules are added or modified to manage the changes in the group membership, and to perform multicast routing, which is supposed to maintain a tree-based multicast infrastructure among the QoS/multicast-capable routers.

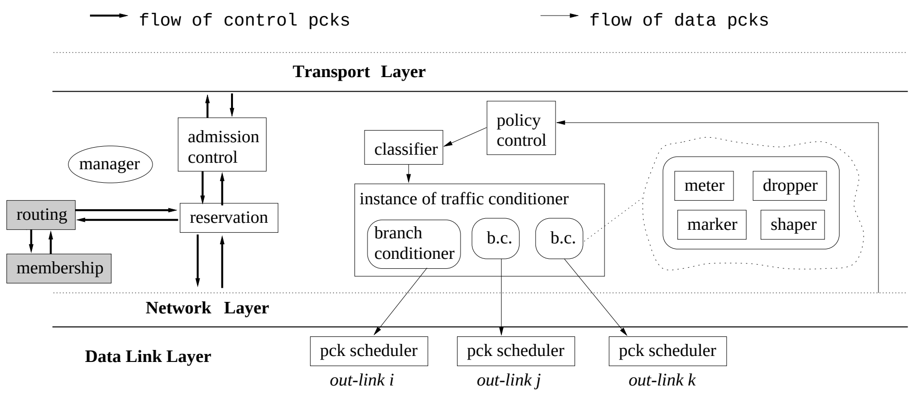


Fig.2. QoS multicast functional architecture

Different receivers may be characterized by different resource availability, thus requiring different levels of QoS. To deal with the receivers heterogeneity, packet processing may be performed according to different policies for different destinations. As the consequence, in a branch router in the multicast tree one copy of the packet management modules (the *branch conditioner* in Figure 2) must be instantiated for each downstream path. These instances must apply appropriate traffic conditioning policies according to the QoS to deliver to their downstream destinations.

For what concerns the end-to-end layer, to support the multicast traffic, the end-to-end modules are required to be group-aware, and must compensate the heterogeneity among different destinations. If *fair delay* requirements are to be satisfied, the modules at each destination must be configured, which carry out the delivery of the data flow at the final receiver. When *reliability* requirements (i.e. *all* destinations receive with a certain probability) or *atomicity* requirements (i.e. *all* or *none* of the destinations receive) must be satisfied, the service becomes much harder to be provided and, in the latter case above, the set of `delivery` modules are requested to execute an agreement protocol.

Further considerations regarding the issues related with multi-domain multicast routing are discussed in [13].

4 The Architecture Implementation

As a preliminary step towards the architecture implementation, we studied the mechanisms that can be adopted to realize the functional modules, in either the IS model or the DS model. We focused on the modules for the service set-up, and we implemented some alternative policies for each of them.

4.1 The Diff-serv Approach

In figure 3 we show the architecture we have considered to support QoS according to the statically configured DS model. In the DS model, the system configuration is in charge of the *bandwidth broker* [12], that corresponds to the routing and reservation modules in the proposed architecture. Applications must notify to their ISP the service they require, the profile of their traffic and the destinations of each microflow. Periodically, the bandwidth broker performs call-admission control, and it computes the paths along which the behaviour aggregates must be forwarded. To this purpose, it exploits the topology and resource availability information collected by a link-state protocol. The bandwidth broker configures the routers that must be traversed by the QoS traffic, so that they implement the appropriate PHBs. We have designed some algorithms that may be used to implement the bandwidth broker. They build a *multicast source-based routing tree* structure along which the behaviour aggregates are forwarded, and have a centralized control. To support multicast services with QoS, algorithms for routing data packets, optimized in order to minimize connection costs only, are not recommendable. It is also necessary to consider constraints regarding the total maximum delay and especially the difference among delays experienced by the packets in reaching their destinations (*fair delay*). Therefore, since the considered problem depends on three different constraints (cost, delay and fair delay), any effort of searching its optimal solution is not effective.

Let us define the maximum source-destination delay Δ_{max} as the age of obsolescence of the packet information. The maximum spread of the source-destination delay δ_{max}, also called fair delay, is the maximum difference of the end-to-end delay for every couple of destinations belonging to the same multicast group. The link cost is defined as the ratio between the allocated bandwidth and the total link bandwidth; this means that the cost increases as the available

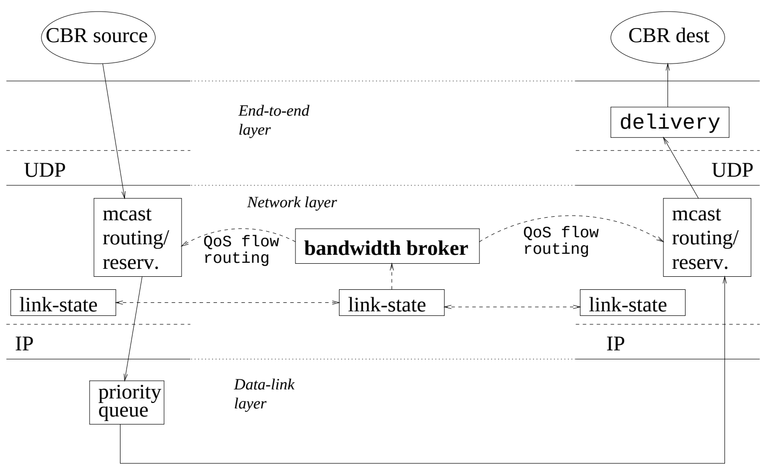


Fig.3. Layout of the architecture implementation for the DS model

bandwidth decreases. Inside a network, let us consider a routing tree T, with cost C, corresponding to a multicast group M with source s. For every node $u \in M$ a route $P(s,u)$ exists in T. Saying r a generic branch of T with cost $c(r)$ and correspondent average delay $D(r)$, the considered constraints can be analytically written as follows:

$$\Delta = \sum_{r \in P(s,u)} D(r) \leq \Delta_{max} \quad \text{(maximum delay constr.)} \tag{1}$$

$$\delta = \max_{u,v \in T} |\sum_{r \in P(s,u)} D(r) - \sum_{q \in P(s,v)} D(q)| \leq \delta_{max} \quad \text{(fair delay constr.)} \tag{2}$$

$$C = \sum_{r \in T} c(r) \leq C_{max} \quad \text{(cost constr.)} \tag{3}$$

where C_{max} is the maximum acceptable cost of a multicast connection. Starting from the knowledge of the three parameters Δ_{max}, δ_{max} and C_{max}, the heuristic algorithms presented in this section have been developed to find sub-optimal multicast routing trees.

The DVMA Algorithm (Delay Variation Multicast Algorithm) [14] has been conceived to build a routing tree satisfying, when possible, the maximum delay constraint and, at the same time, the fair delay constraint. When it fails, the algorithm identifies a multicast tree whose maximum delay complies with constraint (1) at least. Dijkstra's algorithm is encapsulated in DVMA to perform the first attempt path search, by identifying a shortest path tree T_0 minimizing the source-destination delays. If either T_0 complies with both delay constraints (1) and (2), or the maximum delay observed by the farest destination is greater than Δ_{max}, no further computation is performed. The DVMA enhancements, with respect to its Dijkstra's subroutine, are activated when the maximum delay is lower than $\varDelta_{max}$, but the fair delay is greater than δ_{max}. In order to overcome that problem, DVMA operates by lengthening the shortest paths of T_0. DVMA considers the farest destination v, connected to s by an incomplete tree produced by Dikstra. DVMA selects some alternative paths to reach the members of M not belonging to such tree yet. Each of them is joined when the correspondent path complies with both constraints (1) and (2). In this case that path becomes a new branch of the tree. DVMA has been one of the first algorithms devoted to guarantee QoS. On the other hand, it does not take into account the network efficiency, since cost constraints are not included in it.

The CCDVMA Algorithm (Cost Conscious Delay and Delay Variation Multicast Algorithm) [9] has been developed to include connection costs within the goals of QoS multicast routing. CCDVMA uses the Dijkstra's algorithm to identify a first attempt multicast tree, which is computed considering the link costs. Consequently, the resulting paths are characterized by a minimum cost, but they do not optimize the source-destination delay. After that, CCDVMA selects a given number of possible alternative routes, to join the nodes not reached by the tree, trying to satisfy rules (1) and (2). In practice, the algorithm is cost optimized first and delay constrained afterwards. When the two delay limits cannot be simultaneously fulfilled, CCDVMA privileges the reduction of the fair delay, since it is supposed to be the most important QoS parameter.

The CDSAMA Algorithms (Cost and Delay Sensitive Applications - Multicast Algorithm) [4] use a different approach. First of all, CDSAMA chooses a minimum cost tree T by means of an adaptation of the Dijkstra's algorithm operating on the link costs. After that, among all the destinations, CDSAMA selects the one (say node q) characterized by the minimum delay on the path that connects it with the given source along T. Then the algorithm looks for alternative $s \rightarrow q$ paths, not belonging to T but having relatively low costs and delay lower than the relative constraint. Such paths are obtained erasing, one per time, every links present on the $s \rightarrow q$ path in T. For every path characterized in this way, CDSAMA finds the best link connecting a new destination to the tree. The choice is taken according to the following rules:

a) if only one path is characterized by a maximum delay $\varDelta$ with $\varDelta \leq \varDelta_{max}$, it is selected;
b) if multiple paths exist, having a maximum delay lower than, or equal to, $\varDelta_{max}$, the new path is selected as the one that is the best in terms of maximum difference δ (with respect to all the other destinations already reached by the tree);
c) if multiple paths are characterized by either maximum delay $\varDelta \leq \varDelta_{max}$ or maximum difference of delay $\delta \leq \delta_{max}$, the path having the minimum cost is selected.

In this way, node after node, repeating the same procedure illustrated above, CDSAMA provides a multicast tree that connects all the destinations with the given source. In practice, when possible, the algorithm identifies a tree that is compatible with both the delay constraints. Among all the trees that satisfy such limits, it chooses the cheapest. When the algorithm fails, in any case, it suggests the choice of the closest solution to the optimal one. CDSAMA2, a modification of CDSAMA, differs from CDSAMA in the Dijkstra subroutine, that produces the first multicast tree T considering the link delays. Being essentially focused on QoS, CDSAMA2 accepts higher costs as a counterpart of a better performance.

The DVMA2 Algorithm [5] represents a possible evolution of DVMA. It is a 3-constrained multicast algorithm, obtained optimizing the procedure adopted by DVMA to identify alternative source-destination paths. More precisely, with respect to the original algorithm, DVMA2 discards the paths which do not comply with the QoS parameters during their construction, without waiting for their complete definition. In this way, the computational complexity results to be reduced, making this algorithm well suited in a dynamic context, where the modification of the multicast group composition, due to join-and-leave operations, implies a frequent tree re-fitting. The output of the algorithm is represented by a tree minimizing the fair delay, even when the relative constraint (2) is violated.

4.2 The Int-serv Approach

In the IS model, we were interested in studying the impact of both the flows and the group membership dynamics on the end-to-end QoS. Starting from the NS-2 simulation package, we have developed a modular implementation of the described architectural framework, where functional modules can be incrementally added or substituted to observe primitive QoS parameters, such as received data rate, delay, jitter and fair delay, under different traffic sources. The NS-2 simulation package supplies for modules implementing many standard network protocols. At the current state, though, only a template of RTP [15] is available for what concerns QoS support.

The simulation framework currently involves the basic modules for each architectural level; alternative implementations have been realized for some modules. The modules can be variously combined to perform experiments under different system settings. In Figure 4 we show the architecture under test.

As the packet scheduling policies we currently consider:

FIFO: with this policy best effort and QoS packets have the same priority. No resource reservation is performed;

weighted fair queue (WFQ): QoS packets are assigned a weight proportional to the amount of resources reserved for the QoS traffic [7];

worst-case fair weighted fair queue (113:WF2Q): it is a modification of the WFQ, that better approximates the fluidic model, thus improving the fairness in the bandwidth sharing amongst flows [1];

priority (PQ): QoS packets virtually have all of the bandwidth reserved.

The WFQ behaviours depend on two parameters: the *promptness parameter d* [7], and the *weight normalization.* The promptness parameter represents a sort

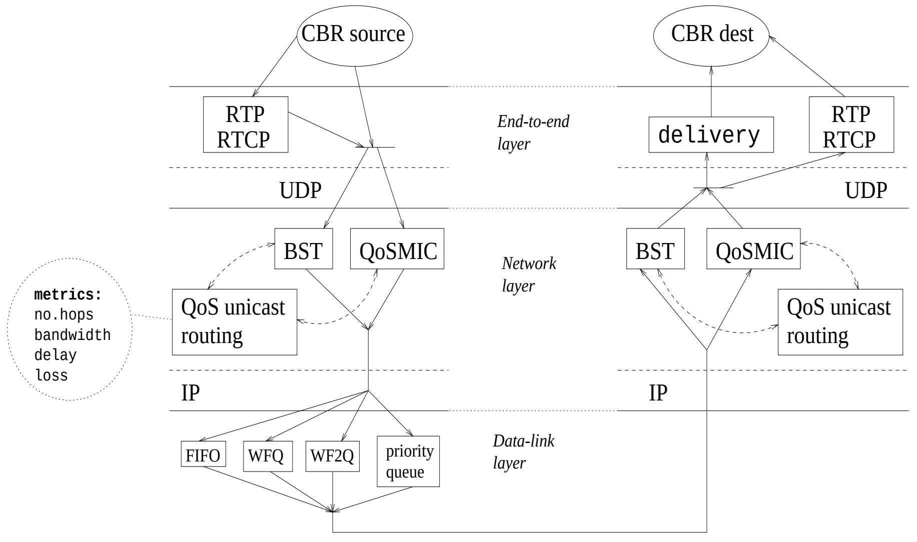


Fig.4. Layout of the implemented architecture

of WFQ memory that remembers when the last packet has been scheduled for each flow. It is used together with the flow weights to compute the transmission time of the packets received at a node: the greater d, the less the low-rate flows are penalized with respect to high-rate flows. Without weight normalization, the bandwidth assigned to a flow f cannot be used by other flows also when no f packet is queued, that is, the policy is non-work-conserving.

NS-2 offers both unicast and multicast routing protocols. As the unicast protocol we consider the Distance Vector. We modified the original class of NS-2 so that paths can be chosen according to several quality metrics: not only distance in hops as usual, but also offered bandwidth, delay or loss probability. Hence, a family of QoS-sensitive unicast routing protocols is available.

The `multicast routing` and `resource reservation` modules are obtained by an implementation of the QoSMIC protocol [8], which has been chosen among others [10,2] for its simplicity and flexibility. QoSMIC builds a shared tree T_G composed of the receivers belonging to a multicast group G; it adopts a distributed approach. When a new router nr joins G, it identifies a subset of the nodes in T_G as its *candidates* to join to the tree. A node n_c candidates if the quality of the unicast path connecting n_c to nr is such that the QoS requested by nr can be provided and, possibly, n_c realizes a local optimum in terms of *quality-of-path*. Several heuristics can be used to characterize the set of candidates; we adopted the *multicast tree search* policy with the *directivity* mechanism for the selection of the candidates. Among them, nr chooses the router (and the branch) that offers the best QoS. QoSMIC can consider only one *quality-of-path* metrics at a time; it requires an underlying QoS-sensitive unicast routing protocol that finds “good” paths according to that metrics. The QoSMIC protocol can be used to provide call admission and resource reservation in both IS and DS domains. So far, we performed experiments in the IS model only. Besides of QoSMIC, other multicast routing protocols are natively available with NS-2, such as Bidirectional Shared Tree (BST), that uses the distance in hops as the quality metrics, and that we used in comparison with QoSMIC. At the end-to-end layer, we modified the NS-2 RTP class to meet the real implementation of that protocol. We interfaced it with the underlying routing protocols.

5 Performance Evaluation

5.1 The Diff-serv Approach

In this section the performance of the algorithms presented in Section 4.2 is analyzed and compared. We tested such algorithms in networks whose topology has been randomly generated. Each node of the network transmits uniformly

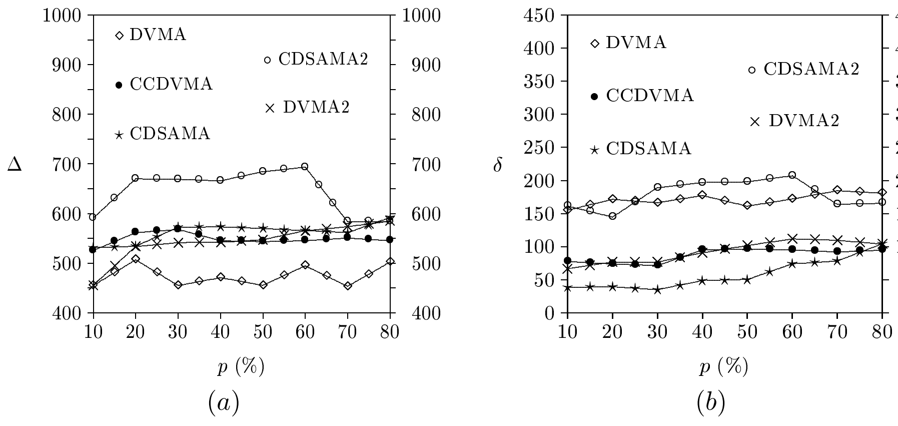


Fig.5. (*a*) Maximum delay towards 2 destinations in a 20-nodes network. (*b*) Maximum fair delay towards 2 destinations in a 20-nodes network

distributed unicast traffic by means of packets of fixed length. The rate of the unicast traffic is determined by the parameter p, that is the probability that a node emits a unicast packet in a time slot. In the sequel, p varies from 10% to 80%. A multicast connection between a source and a multicast group is then overlapped on the unicast communications. The positions of source and destinations of the multicast connection are randomly selected. We considered networks composed by 20 and 30 nodes, supporting a network load equal to $20p$ and $30p$ respectively. For every class of networks, simulations have been repeated for different sizes of the multicast group. In particular, the membership is equal to 10%, 20% and 30% of the total number of nodes belonging to the network. For every simulation conditions, Δ, δ and C are measured. The parameters Δ and

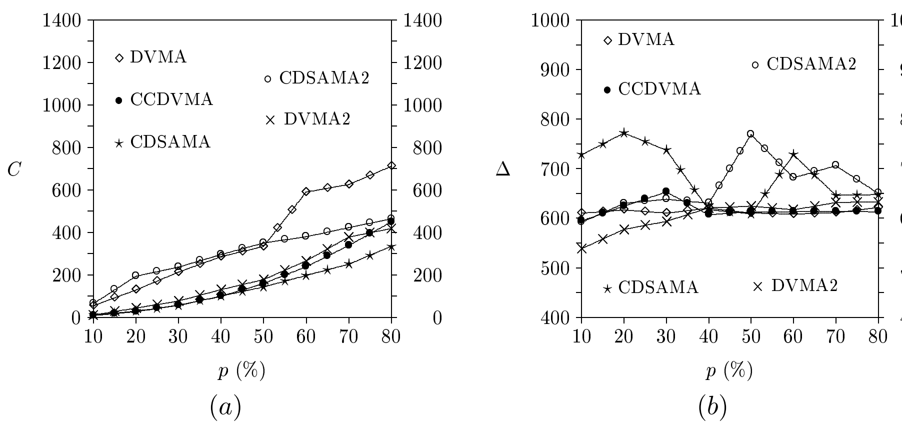


Fig.6. (*a*) Cost of the tree connecting 2 destinations in a 20-nodes network. (*b*) Maximum delay towards 4 destinations in a 20-nodes network

δ are measured in tens of time slots, whereas C is reported in centesimal units. In Figs. 5, 6 and 7 we report the graphs relative to networks composed by 20 nodes, where the multicast group size is set equal to 2 and 4. Similar results have been obtained with 6 destinations.

From this group of plots we can summarize the following considerations:

- DVMA, being constrained by delay specifications only, is characterized by a low efficiency, whereas it provides good QoS with respect to the delays;
- CCDVMA, with respect to DVMA, shows higher values of δ and Δ but, obviously, it is preferable for the cost profile;
- CDSAMA has an optimal behaviour in terms of connection efficiency; its classic behaviour (characterized by high Δ and δ values) is not evident here because of the small network dimension;
- CDSAMA2 is the opposite of CDSAMA, that is, it has acceptable QoS levels (especially when the group size increases) at the expense of higher tree costs;
- probably DVMA2 provides the best cost/performance ratio, since it shows a quite good behaviour, at a cost similar to that of the other algorithms, but always lower than the DVMA one.

We repeated the same experiments in the case of 30-nodes networks, where the multicast group is composed by 3, 6 and 9 destinations. In those cases, we observe that all the algorithms are characterized by behaviours similar to those of the 20-nodes case, where the worsening of the performance is essentially due to the increase of the network dimension. Even though DVMA is focused on delay minimization only (no cost constraint is considered), it fails in determining the trees having the minimum δ for every intensity of the unicast traffic. In fact in general our new algorithms, which instead operate on three constraints, show a better behaviour. For the same reason, the DVMA efficiency is always worse than that of the other algorithms (CDSAMA2 excepted). In particular,

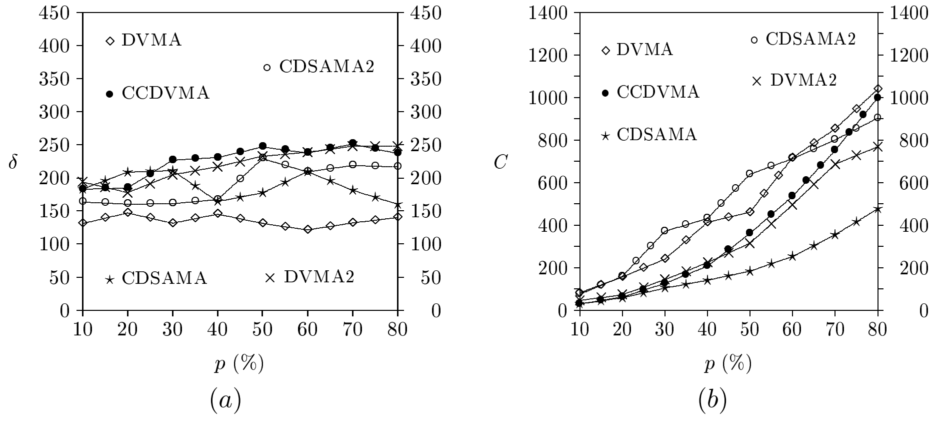


Fig.7. (*a*) Maximum fair delay towards 4 destinations in a 20-nodes network. (*b*) Cost of the tree connecting 4 destinations in a 20-nodes network

the correspondent tree cost increases as load gets higher. Among the developed algorithms CDSAMA2, essentially but not exclusively focused on delay optimization, performs particularly better than DVMA. Finally we can observe that, for every algorithm, all the QoS levels seem to be affected by load increase very weakly. This is not true for the cost parameter.

5.2 The Int-serv Approach

We performed measures with the protocol stack described in section 4.2 under different network conditions. We consider a meshed network of 64 routers inter-

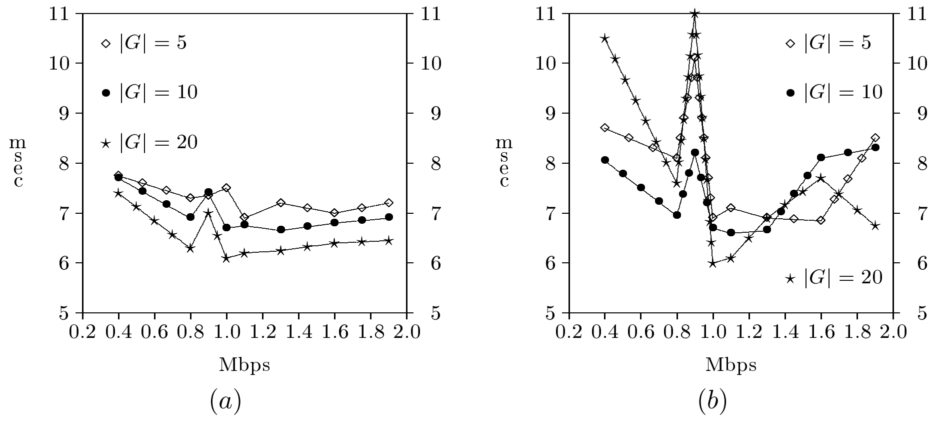


Fig.8. (*a*) Average end-to-end delay with PQ. (*b*) Average end-to-end delay with 113:WF2Q

connected by optical fiber links having a 2 Mbps bandwidth and length variable between 50 and 100 Km. QoS CBR traffic originates in the tree root; we performed our measures with different transmission rates. The background, best effort traffic is uniformly distributed all over the network and the average use of the bandwidth resources is 33%. We considered group memberships of 5, 10 and 20 nodes. As the quality of path metrics we considered the bandwidth.

QoSMIC has proved to efficiently perform call admission: when the sum of the QoS source rates approximates the link capacity, the receivers in the intersection of the group memberships successfully join the tree only for a subset of those sources. Unfortunately, the efficacy of the heuristics used by QoSMIC to characterize a set of candidates and distribute the load is limited by the dependency on the unicast routing. The unicast routing provides the paths having the maximum bandwidth, according to the link characteristics. Along those paths all the flows are forwarded. When links saturate along those paths, destinations have to renounce to receive QoS traffic, although alternative paths with enough resource availability could exist.

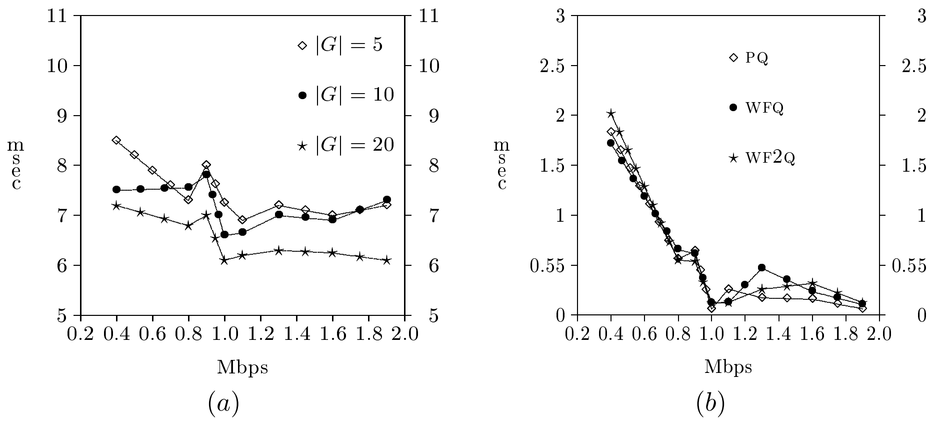


Fig.9. (*a*) Average end-to-end delay with WFQ. (*b*) Jitter for $|G| = 10$ and different packet scheduling policies

We measured the end-to-end QoS achieved with different packet scheduling policies. Obviously, the FIFO policy does not offer any guarantee. The WFQ policy guarantees a better fairness among different flows than WF^{2}Q, only when the prompteness parameter is set to 1 and no weight normalization is performed. This way, the queueing delay of the slow flows is bounded. Since CBR sources are honest (i.e., they produce traffic at the claimed rate), PQ does not delay packets in the queues. A packet arriving at a node must wait at most for a packet transmission time before being forwarded. By contrast, the policies aiming at providing a better fairness guarantee the unreserved bandwidth to best effort packets, that are always present, thus worsening the delay experimented by QoS packets and introducing irregularities in the flows.

This is confirmed by the end-to-end delay curves for increasing QoS rate and different group G cardinalities (figures 8 and $9(a)$). Those curves have been obtained by considering two QoS sources: the former has a 1 Mbps rate, the rate of the latter varies between 0.4 Mbps and 1.9 Mbps. The performance indexes reported by these and the following figures concern the flow generated by the latter source. The WFQ policy has $d = 1$ and no weight normalization. The impact of the queueing delay is particularly evident when the network is congested (rate of the second source of 0.9 Mbps). For higher rates the contention between the two sources disappears, as their flows cannot traverse the same links.

Obviously, PQ does not behave well when irregularities are introduced in the generated CBR flows: bursts of a flow are drained by stealing bandwidth to the flows with lower priorities, thus violating the flow isolation. As a matter of fact, those delays are negligible; we observed that all the policies guarantee the correct traffic arrival rate at the destinations grafted to the tree. The fair delay behaviour is consistent with that of the end-to-end delay: the recipients farer from the source suffer greatly for the queueing delays when WFQ or 113:WF2Q are used. In figure $9(b)$ the jitter is shown, averaged on the recipients of the second source. All the considered scheduling policies are unable to control the jitter, due to the interleaving among different flows at the intermediate nodes. In fact, the jitter goes to zero for high QoS rate just because the best effort traffic is almost completely excluded from the network.

6 Concluding Remarks

In this paper we propose a functional architecture to support QoS in multi-domain IP networks. We describe two implementations of the architecture, to support multicast QoS in both the int-serv model and the diff-serv model. We discuss the performance results obtained with those implementations.

For the DS model, we studied some heuristic methods of path searching submitted to three cost/performance constraints for meshed topology. In particular the new CDSAMA, CDSAMA2 e DVMA2 algorithms have been presented and tested in comparison with the previous DVMA and CCDVMA solutions. The simulation results show that the new techniques offer a maximum delay slightly greater than that produced by DVMA. Similarly, encouraging results characterize the performance of such algorithms with regard to the fair delay as well. On the other hand, significant advantages are achieved in terms of connection costs.

For the IS model, some interesting considerations derive from the obtained results, regarding the policies that seem more promising to realize QoS support. Jitter and fair delay can be controlled only at the end-to-end layer, by appropriately configuring the delivery modules at the recipients. All the scheduling policies guarantee the arrival rate. But, with respect to the end-to-end delay, the PQ policy behaves better. To avoid the negative effects of irregularities in the flows, *traffic shaping* modules should be implemented at the source nodes, and possibly at the intree routers.

A lot of work still remains to be done. We are extending the framework with other policies either available in NS-2 (e.g., RED, CBQ) or that we are currently implementing, such as `Weighted Round-Robin`. We are modifying QoSMIC to adopt it in the DS model as a distributed bandwidth broker, to support dynamic system reconfiguration. We are implementing both the QoS CBT protocol [10] and the QoS PIM-SM protocol [2].

References

1. Bennet, J.C.R., Zhang, H.: WF^{2}Q: Worst-case Fair Weighted Fair Queueing. Proc. IEEE Infocom'96 (1996) 120–128
2. Biswas, S., Izmailov, R., Rajagopalan, B.: A QoS-Aware Routing Framework for PIM-SM Based IP-Multicast. nternet Draft, draft-biswas-pim-sm-qos-00.txt. Work in progress (1999)
3. Blake, S., Black, D., Carlson, M., Davies, E., Wang, Z., Weiss, W.: An Architecture for Differentiated Services. Internet Draft, draft-ietf-diffserv-arch-02.txt. Work in progress(1998)
4. Borella, A., Cancellieri, G.: Improving the Cost/Performance Ratio in Multicast Communications. Proc. NOC 2000 (2000) 187–190
5. Borella, A., Cancellieri, G.: Remote Time-Alignment of Interactive Services through Efficient Multicast Algorithms. Proc. ECUMN 2000 (2000) 239–246
6. Braden, R., Clark, D., Shenker, S.: Integrated Services in the Internet Architecture: an Overview. RFC 1633. Work in progress (1994)
7. Demers, A., Keshav, S., Shenker, S.: Analysis and Simulation of a Fair Queueing Algorithm. Proc. ACM SIGCOMM'89 (1989) 1–12
8. Faloutsos, M., Banerjea, A., Pankaj, R.: QoSMIC: Quality of Service sensitive Multicast Internet protoCol. Proc. SIGCOMM 1998 (1998)
9. Haberman, B., Rouskas, G.N.: Cost, Delay and Delay Variation Conscious Multicast Routing. Proc. INFOCOM 2000 (2000)
10. Hou, J., Tyan, H.-Y., Wang, B.: QoS Extension to CBT. Internet Draft, draft-hou-cbt-qos-00.txt. Work in progress (1999)
11. Nichols, K., Blake, S., Baker, F., Black, D.: Definition of the Differentiated Services Field (DS Field) in the IPv4 and IPv6 Headers. Internet Draft, draft-ietf-diffserv-header-04.txt. Work in progress (1998)
12. Nichols, K., Jacobson, V., Zhang, L.: A Two-bit Differentiated Services Architecture for the Internet. Internet Draft, draft-nichols-diff-svc-arch-00.txt. Work in progress (1997)
13. Pagani, E., Rossi, G.P.: A Functional Architecture for End-to-end Quality-of-Service in a Multi-domain Network. Proc. IEEE ECUMN 2000 (2000) 20–34
14. Rouskas, G.N., Baldine, I.: Multicast Routing with End-to-end Delay and Delay Variation Constraints. IEEE Journal on Selected Areas in Communications, Vol. 12, No. 3 (1997) 346–356
15. Schulzrinne, H., Casner, S., Frederick, R., Jacobson, V.: RTP: A Transport Protocol for Real-Time Applications. RFC 1889. Work in progress (1996)
16. Zhang, L., Braden, R., Berson, S., Herzog, S., Jamin, S.: Resource reSerVation Protocol (RSVP) - Version 1: Functional Specification. RFC 2205. Work in progress (1997)

Resource Allocation and Admission Control Styles in QoS DiffServ Networks[⋆]

Mario Gerla[1], Claudio Casetti[2], Scott Seongwook Lee[1], and Gianluca Reali[3]

[1] Computer Science Dept., UCLA
Los Angeles, CA, U.S.A.
{gerla, sslee}@cs.ucla.edu
[2] Dipartimento di Elettronica,
Politecnico di Torino, Italy
casetti@polito.it
[3] Dipartimento di Ingegneria Elettronica e dell'Informazione,
Universita' degli Studi di Perugia, Italy
reali@diei.unipg.it

Abstract. In this paper we propose a DiffServ architecture for the support of real time traffic (e.g., video) with QoS constraints (e.g., bandwidth and delay) over an IP domain. The main goal of the paper is to identify solutions which provide QoS guarantees without requiring per flow processing in the core routers (as is commonly done in IntServ solutions) and which are thus scalable. We propose, and evaluate through simulation, different approaches for call admission control (CAC) and resource allocation. These approaches are all consistent with the DiffServ model, but place different processing and signaling loads on edge and core routers. Paths are computed by means of a QoS routing algorithm, Q-OSPF, and MPLS is used to handle explicit routing and class separation.

1 Introduction

This paper deals with the DiffServ approach by which Internet network service providers can offer different service quality to different classes by using different admission control and reservation "styles." We recall that the main goal of the DiffServ approach is to overcome the well known limitations of Integrated Services and RSVP, namely, low scalability of per-flow management in core routers. The basic approach of DiffServ is to manage traffic in core routers by applying different per-hop behaviors (PHBs). PHBs are specified by a code, named DS codepoint, in a dedicated field in the header of IPv4 and IPv6 datagrams. This way, traffic flows can be aggregated into a relative small number of PHBs, which can be easily handled by core routers without scalability restrictions.

In this context, we seek scalable admission control and reservation solutions for DiffServ over an IP domain. The main contribution of this work is the definition of alternate connection admission control (CAC) strategies which provide

[⋆] This research was supported in part by grants from CISCO, NASA, NSF and the State of California a variable degree of QoS guarantees to behavior aggregates. Another key performance to watch in the proposed solutions is the level of involvement of core routers in traffic management. The intent is to move the complexity and the processing O/H from the core to the boundary of the DS domain. The guidelines of our proposal are compatible with [1], and in particular include only aggregated classification state within the core routers. However, due to space limitations, in this paper we will not address all the issues raised in [1]. Rather, we will describe and analyze only those issues related to performance guarantees. In regard to the end-to-end behaviors, our attempt is to provide hard and soft bounds on delay, and loss. These are the same targets of the guaranteed quality of service specified in [2]. Our approach departs from that in [2] in that we simplify core router by eliminating re-shaping of traffic in core routers. This enhances scalability since reshaping is done on a per flow basis. While the lack of reshaping implies a progressive alteration of the statistical properties of each micro-flow along the path, some QoS guarantees are still provided to each micro-flow while operating only on behavior aggregates in core routers. Depending on the processing and signaling load that we are willing to tolerate in core routers, we define three alternate CAC approaches, discussed in section 3. In all three cases, paths are determined by means of a QoS routing algorithm. We assume that the forwarding layer of the protocol stack is based on MPLS [3], so that the DiffServ Behavior Aggregates can be mapped into Label Switched Paths. The above CAC styles are characterized by different levels of complexity and provide different levels of QoS guarantees, as discussed in the following sections. We also study the coexistence of different styles in the same network and show that WFQ is necessary to maintain "differentiated" performance in the different classes. The various approaches are illustrated and validated via simulation using a PARSEC wide area network simulation platform. The real-time traffic is represented by MPEG video traces. In the following sections we introduce the various "building blocks" of the proposed DiffServ architecture starting from its foundation, namely QoS routing.

2 QoS Routing

The QoS routing scheme plays a key role in this architecture. It serves the dual purpose of finding feasible routes as well reporting (to each edge router) the quality of an existing route. A general QoS routing problem consists of finding a path from a source to a destination, which is suitable to a given application with given end-to-end constraints. The importance of QoS routing for supporting QoS in the Internet is testified by the IETF activity in this field [4]. In our framework, the constraints that we will consider are delay and bandwidth, as they are particularly suited to real-time applications like IP telephony and video conferencing.

An *optimal path* is here defined as a path with minimum number of hops which still satisfies a set of multiple constraints. In general, optimal routing with multiple metrics is known to be an NP-complete problem [5]. To avoid the complexity of exact, combinatorial solutions, several papers [6,7,8] have addressed QoS-constrained routing problems using a variety of heuristic strategies. An interesting departure from the current heuristics is offered by [9], where a QoS routing algorithm based on Bellman-Ford is proposed, the Multiple Constraints (MC) Bellman-Ford, which finds "optimal" (min hop) solutions when dealing with bandwidth and delay constraints.

In our experiments, we have restricted our study to intradomain routing and have used as a starting point OSPF, an intradomain routing protocol. We have extended OSPF (calling it Q-OSPF) to enable QoS routing by including available link bandwidth and packet delay in the link state packet. Although the standard OSPF uses the Dijkstra algorithm for path computation, in Q-OSPF such task is performed by the MC Bellman-Ford algorithm, which is instrumental in the support of measurement-based Admission Control schemes.

3 Call Admission Control

Next to QoS routing, call admission control is the most critical function in networks supporting real time traffic. In this section we review several CAC strategies, while we refer the reader to [20] for implementation issues.

3.1 Measurement-Based CAC

Using measurement-based strategies, such as the ones described in [10,11,12,13], entails periodically collecting samples of meaningful quantities representing the state of the network, such as the bandwidth available on a link. Admission decisions are then based on these measurements rather than on worst-case bounds. It is a well known fact that, while measurement-based algorithms can achieve a higher network utilization, they can only provide *statistical* guarantees, and are therefore practical only for highly predictable traffic, such as voice connections, and for large traffic aggregations. If *deterministic* delay and loss guarantees need to be achieved, a measurement-based admission scheme falls short of the task. In this paper, we test the pros and cons of using Q-OSPF as a collection vehicle for timely traffic measurements. In our simulation experiments we will in fact examine strategies where MPEG video streams are subject to measurement-based admission control.

With this approach, the edge router performs Connection Admission Control (CAC) on the incoming video calls solely based on the bandwidth and delay information provided by Q-OSPF. These parameters are monitored periodically by each router using a measurement window and are packed and distributed to all the other nodes via the Q-OSPF Link State Update packet. Since timeliness of these measurements is crucial, we trigger a new Link State packet when the measurement changes exceed a specified percentage; and at the same time reduce the Link State Update interval from 30 minutes (the current Internet practice) to 2 seconds, as long as changes persist. Upon receiving a call request from the subscriber network, the edge router inspects one of the current active MPLS paths to determine if there is enough residual bandwidth BR to accommodate the call and if the delay constraint is met. If the answer is affirmative, the call is immediately accepted. If the existing MPLS pipes are saturated, the router invokes a new QoS route computation. If a new feasible path is found, the call is forwarded onto such path. Else, the call is rejected. As mentioned earlier, a path is feasible if it satisfies the end-to-end delay constraint, and if it has residual bandwidth $\geq BR$. The choice of the residual bandwidth BR is dictated by the need to guarantee adequate performance to the new coming connection and, at the same time, protect the performance of ongoing connections from the statistical fluctuations of the real time traffic around the average value measured by Q-OSPF. Note that the optimal path dynamically changes as a function of load level and load distribution in the network. This is because metrics depend on traffic loads (e.g., bandwidth, etc). In our case, the selected path is pinned at call setup time by MPLS [3,14].

3.2 Resource Allocation-Based CAC

As discussed in [20], real-time, high-priority traffic classification is performed according to a set of traffic descriptors (r_s, P_s, B_{TS}). These descriptors are determined by a characterization of the traffic by means of a Dual Leaky Buffer (DLB) traffic shaper. Specifically, r_s indicates the rate at which the leaky buffer is fed, while P_s and B_{TS} are the peak output rate and the buffer size, respectively. A further organization in subclasses is performed according to the value of the delay bound D_{max}. This classification is used to store state information management within core routers for each class. Note that this “class” state stored in core routers depends only on the number of classes; it is independent of the number of actual flows in the class. Using the approach described in [20], we can compute, for each flow belonging to class k, an “equivalent capacity” $c_{0k} = f(r_s, D_{max}, B_{TS}, C)$ (with C equal to the channel capacity) and an “effective buffer” b_{0k}. Let us now assume that each core router keeps track of the number of flows currently being transmitted on its outgoing trunks for each subclass. From this number, it can easily compute the aggregate effective capacity allocated so far. It is thus able to determine if an incoming request can be accepted. Moreover, to avoid time-consuming trial and error attempts to route new flow requests on links which may be already saturated, it is desirable to base the routing decisions on the available capacity of each individual link.

To illustrate the Resource Allocation-based CAC procedure, let us consider a set of n subclasses $S_1, S_2, \ldots, S_n$. The state information table at each router consists of the following entries:

$$[OUTPUTTRUNK; AC; NS_1; NS_2; \ldots NS_n]$$

where the entry AC is the available trunk capacity, and $NS_k, k = 1, \ldots, n$, is the number of additional flows that the router can accept for subclass S_k within the specified QoS parameters. For a generic trunk i, j, the corresponding available capacity AC_{ij} is calculated as follows:

$$AC_{ij} = C_{link,ij} - \sum_{k=1}^{n} n_{dk} c_{0k} \tag{1}$$

where $C_{link,ij}$ is the trunk capacity of the link, c_{0k} is the effective capacity of each flow belonging to the subclass k and n_{dk} is the number of flows of that subclass currently being delivered. This parameter is denoted as the aggregation factor, and is known by the router. The number of additional flows $S_k, k = 1, \ldots, n$, is computed using the relation

$$NS_k = \min(\frac{AB_{ij}}{b_{0k}}, \frac{AC_{ij}}{c_{0k}}), \quad k = 1, \ldots, n \tag{2}$$

AB_{ij} being the available buffer space, that is

$$AB_{ij} = B_{ij} - \sum_{k=1}^{n} n_{dk} b_{0k} \tag{3}$$

where B_{ij} is the output buffer size to trunk j and b_{0k} is the effective buffer of each flow belonging to the subclass k.

The computation of all entries of the state information table requires that the number of flows n_{dk} be known. If signaling is performed in conjunction with CAC, the information is immediately available to all routers along the path, since they know exactly how many connections have been accepted so far into the network. They also know how many have been cleared because of explicit signaling. This way, each router knows the aggregation factor n_{dk} for each subclass and can use this information to perform a CAC.

From the implementation point of view it is worth noting that the number of state entries depends uniquely on the number of classes, rather than on the number of flows. In this sense, the approach is very scalable.

The following steps describe the CAC algorithm in more detail. CAC is triggered by a request to an edge router of delivering a new flow belonging to a specific class, say k. The edge router identifies a potentially feasible MPLS path (among the ones available to this class). The request is then sent on the MPLS path to the receiver. Resource allocation is carried out on the way back. Namely, the last router on the path decides if the request can be accepted according to the bandwidth and buffers available, as described above. If the request is not accepted, the source edge router eventually learns about it, and searches for a new path using Q-OSPF. If still no feasible path is found, a reject notification is sent back to the source.

3.3 Hybrid CAC

The Resource Allocation-based CAC enforces accurate performance bounds at the expense of network utilization as well as increased processing load and specialized software in the core routers due to bookkeeping. This load at the core routers can be shifted to the edge routers by making use of Q-OSPF relayed measurements. In fact the edge router can directly estimate the number of flows (in a given class, on a given trunk) from the measured trunk load (for that class) by dividing the load by the sustainable flow rate for the class in question. The larger the number of flows, the more accurate the estimate. More precisely, the following "hybrid" CAC scheme is proposed:

1. compute the number of flows from trunk load measurements
2. multiply the number of flows by the equivalent bandwidth to obtain the current aggregate "equivalent" bandwidth allocation to that trunk
3. perform the CAC functions at the edge router based upon the total available bandwidth and currently available buffer information provided by Q-OSPF, much in the same way as done in the Resource Allocation-based CAC at each core router. This method should lead to results close to the explicit signaling method as long as the number of flows is large so as to mask imprecision due to traffic fluctuations.

4 Simulation Results

Based on the schemes discussed in the previous sections, simulation experiments were carried out to evaluate the performance tradeoffs of the three CAC schemes.

The simulation topology used in these experiments is relatively simple (see Fig. 1). It contains a single bottleneck between sources and destinations. All the simulation experiments were performed with a DLB at every traffic source. The detailed configuration parameters are described in Table 1. The simulation environment is based on the parallel simulation language PARSEC developed at UCLA [18].

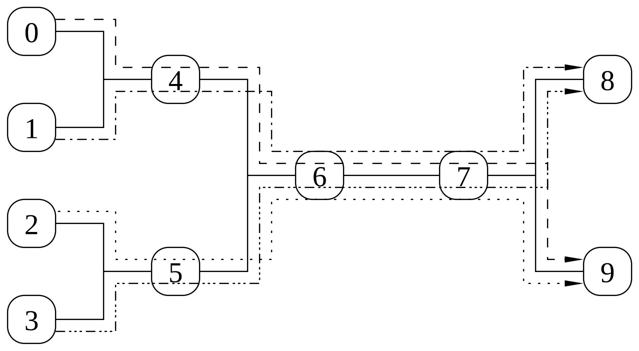


Fig. 1. Simple 10-node topology used to demonstrate the CAC mechanisms and differences.

Table 1. Simulation configuration for the demonstration of the three CAC mechanisms and differences.

Traffic type	MPEG video trace
Traffic average rate	0.7 Mbps
Traffic peak rate	4.5 Mbps
Traffic source	Node 0, 1, 2, and 3
Traffic destination	Node 8 and 9
Link capacity	45 Mbps for all links
Link propagation delay	0.1 ms for all links
Router buffer space	562500 bytes for all nodes
Connection request arrival	1 per second at each source
Connection duration	60 sec of exponential dist.
Bandwidth allocation	1 Mbps
Buffer allocation	12500 bytes

Each connection lasts about 60 seconds on average; the connection requests arrive at 1 per second per source; thus, the bottleneck can carry on the order of 40 to 80 connections. The network has reached steady state after the first 60 seconds. Fig. 2 shows traffic load as a function of time on the bottleneck link between node 6 and 7. As described in the previous sections, the three CAC schemes have quite different characteristics and are expected to give different results. RA-CAC performs most conservatively (lowest throughput and number of accepted connections), but enforces all the constraints (delay and packet loss). M-CAC on the other hand does not always satisfy the constraints, in spite of the fact that the DLB shapers are active at all sources. Recall that M-CAC simply takes the current load measurement carried by OSPF and interprets it as the "allocated" bandwidth. This estimate is extremely optimistic and "aggressive." In particular, it makes no assumptions on the statistical nature of the measured traffic. Consequently, the throughput and number of accepted calls is much higher than for the RA-CAC case. However, traffic fluctuations caused by a change in the background scene on the video trace, for example, can easily lead to large queueing delays and buffer overflow, with packet loss and delay constraint violations as shown in Table 2 and Fig. 4, respectively.

H-CAC performance is very similar to RA-CAC, as expected, since we are dealing here with a relatively large number of video calls in the bottleneck trunk. Thus, the law of large numbers applies, and the number of ongoing calls can be estimated with reasonable accuracy from traffic measurements. The throughput and number of calls accepted by H-CAC is slightly larger than for RA-CAC, possibly because H-CAC underestimates the number of calls on a trunk. This behavior deserves further investigation. Fig. 4 shows the delay distributions for the three schemes and clearly exposes the delay violations of M-CAC. Fig. 4 actually shows a spike for the M-CAC delay at 100 ms. This is because only the packets actually received contribute to the delay distribution. Dropped packets are not accounted for. Given that the buffer size at the bottleneck is 562 kB, the maximum delay at the bottleneck computed according to the methodology in [20] is 100 ms.

In our experiments we have also monitored the line overhead introduced by Q-OSPF. This is an important factor in the overall CAC strategy evaluation

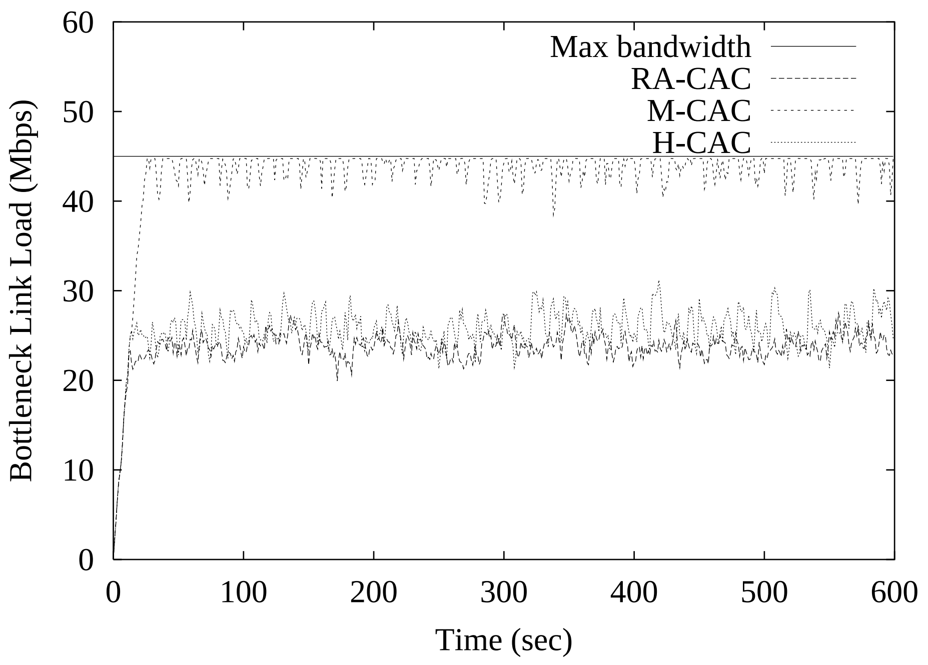


Fig. 2. Bottleneck link load of each case of the three CAC schemes.

Table 2. Performance statistics of the three CACs.

Scheme	# of conn. tried	# of conn. admitted	% of pkt. lost
RA-CAC	2400	465	0 %
M-CAC	2400	864	0.39 %
H-CAC	2400	504	0 %

since Q-OSPF is strictly required by M-CAC and H-CAC, which use Q-OSPF measurements to determine acceptance/rejection at the edge router. In RA-CAC, on the other hand, the link state updates provided by Q-OSPF are not strictly required (as long as it is deemed sufficient to use min hop paths). Or, if Q-OSPF is used for alternate path routing, the update frequency can be drastically reduced since a separate signalling protocol verifies the constraints router by router along the path.

In our Q-OSPF evaluation via simulation, we measured the total amount of OSPF packets transferred through the bottleneck link over the entire simulation time. A total of 2236800 bits during 600 seconds were transferred through the bottleneck via OSPF flooding. Therefore, merely about 3.7 Kbps out of the 4.5 Mbps bottleneck bandwidth were consumed.

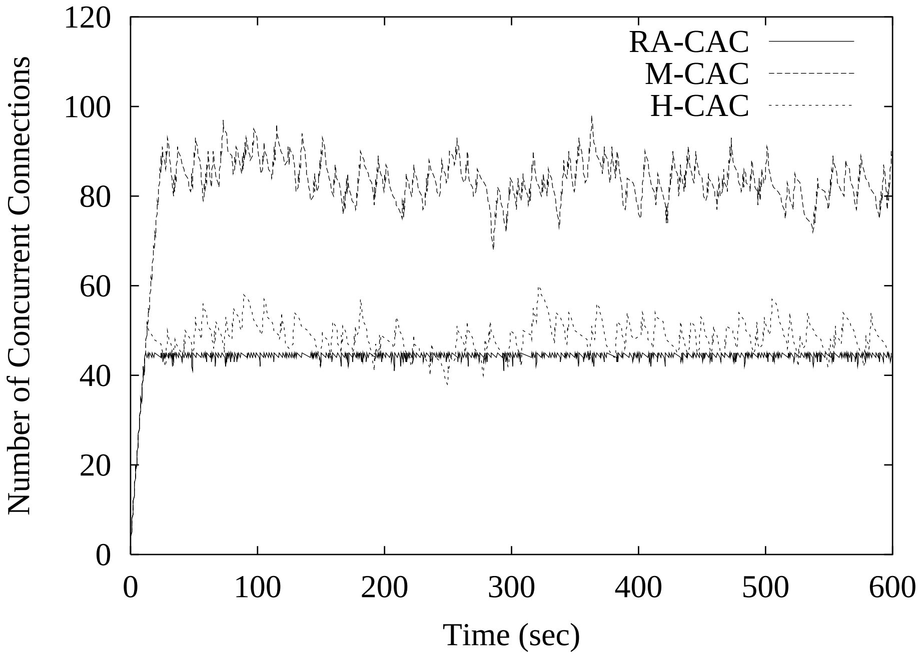


Fig. 3. The number of concurrent connections admitted by the three CAC schemes.

In [19], the overhead analysis was studied in more detail. Those experiments where consistent with ours, which use a 2-second interval. Moreover, the topology is a perfect grid network which is more dense than the topology used in our simulations (thus, more Link State updates). The OSPF overhead we have measured over the grid topology was of little more than 40 Kbps for 36 nodes and of 100 Kbps for 81 nodes, confirming the fact that the O/H impact is negligible up to fairly large network sizes. The chosen update rate (2 seconds) strikes a good balance between network performance and bandwidth overhead [19], and is consistent with previous research results on OSPF QoS routing [4].

In the previous experiments, the different CAC strategies were tested separately. It is conceivable that an ISP provider will offer the choice of multiple CAC "styles," for different prices and different performance guarantees. Some users may tolerate the performance degradation of M-CAC (at heavy load) in exchange for a discount rate. Thus, different styles may coexist in the same network. In the following set of experiments we study the multiple CAC and mixed flow situation.

We define two classes of video users and let them share the network simultaneously, applying a different CAC scheme to each. The problem with this approach is that M-CAC is more aggressive than RA-CAC (it accepts calls even

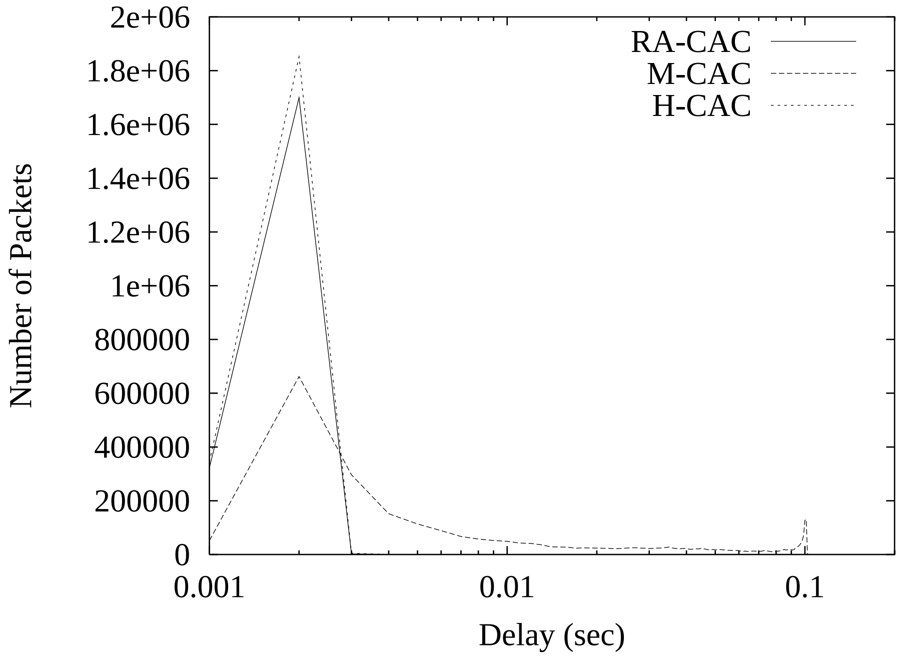


Fig. 4. End-to-end packet delay caused by the three CAC schemes.

when RA-CAC rejects them) and would capture the entire bandwidth. Thus, a proper fair scheduling scheme (e.g., WFQ) is required to "protect" the RA-CAC class from the M-CAC class.

The new simulation scenario is shown in Fig. 5; the configuration parameters are listed in Table 1 and 3.

We deploy 3 queues: control, RA-CAC and M-CAC. The weights for the queues determine the maximum bandwidth allowed for each queue in the case where all queues were fully loaded. WFQ achieves max-min fairness: if one queue is not fully consuming its maximum bandwidth other queues use the leftover bandwidth.

Having assigned weight = 5 to queue 0 for (control packets) and 20 to the other two queues (RA-CAC and M-CAC packets), the corresponding bandwidth assigned to the queues at the bottleneck links is 5 Mbps for the control packet queue, and 20 Mbps each for RA-CAC and M-CAC queues. As in the previous simulation scenarios, we consider two traffic situations - low and high.

Fig. 6 shows the number of concurrent calls in the network. Since the queue of RA-CAC has maximum available bandwidth of 20 Mbps, the maximum number of concurrent calls is 100, which is (20 Mbps / 0.8 Mbps) $\times$ 4 possible paths in the bottleneck. As expected, M-CAC allows more calls due to stale link state information and optimistic traffic load estimates.

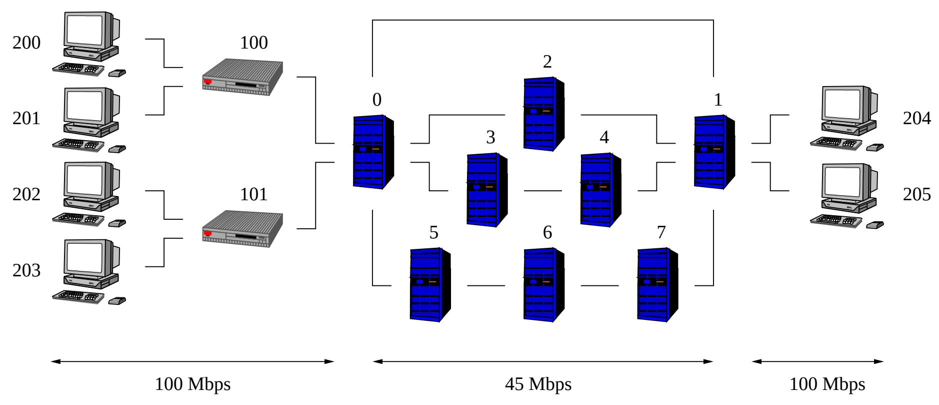


Fig. 5. Simulation topology with multiple alternate paths at the core.

Table 3. Simulation configuration of the second scenario.

Number of queues	3 (0 through 2)
Queue weights	5 for queue 0 which is primarily for control packets such as Q-OSPF LSA packets. 20 for queue 1 which admits calls by RA-CAC. 20 for queue 2 which admits calls by M-CAC.
Traffic generation	In lightly loaded case, calls arrive at one of the source nodes every 4 seconds for each queue, 1 and 2. In highly loaded case, calls arrive at one of the source nodes every 0.5 second for each queue.

Fig. 7 shows that the M-CAC streams achieve their limit, namely 20 Mbps. In fact, no longer bounded by a physical limit (i.e., 45 Mbps), the M-CAC traffic is now able to even surpass the 20 Mbps soft limit and in part invade the bandwidth left over by RA-CAC. As shown in Fig. 8, the RA-CAC traffic achieves only 15 Mbps on the bottleneck links.

Remarkably, the bandwidth "borrowed" from RA-CAC significantly improves M-CAC end-to-end delay performance. The delay distributions clearly show this trend in Fig. 9. Table 4 summarizes the CAC statistics of the simulation results. We note that even the M-CAC traffic is now in compliance of the 100 ms delay constraint. The explanation is that M-CAC has now at its disposal more bandwidth than advertised by the available bandwidth measurements. Thus, the optimistic estimation is now rewarded. As a general rule, the presence of RA-CAC streams ensures that there is still extra bandwidth statistically available on the link even when the RA-CAC portion of the bandwidth bas been fully allocated. Aggressively, optimistic CAC strategies (such as M-CAC) can take advantage of such extra bandwidth and avoid excessive delay degradation.

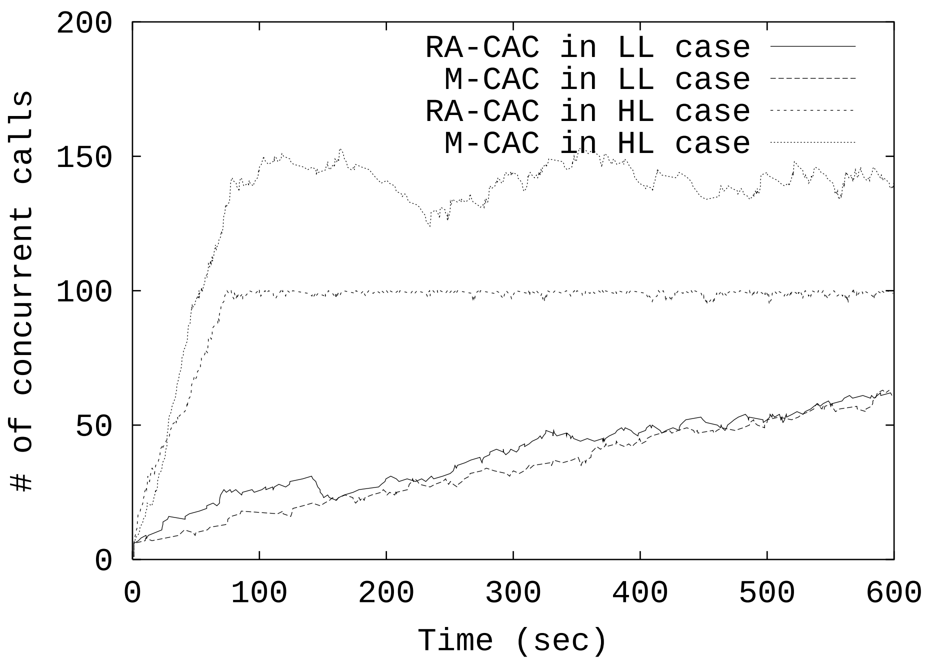


Fig. 6. The number of concurrent calls when each CAC is deployed with different traffic load. (LL = lightly loaded, HL = highly loaded)

Combining RA-CAC and M-CAC styles in the same network, together with the use of WFQ offers a number of new options and opportunities. For example, as the number of traffic classes grows, it may become impractical to measure and propagate traffic volume and available bandwidth information for each class. To reduce Q-OSPF overhead and enhance scalability we can conceive a scheme where all the M-CAC traffic classes are "aggregated" into one common group for resource monitoring purposes. Minimum resource allocation to each class on each network link is still guaranteed by per class WFQ. Call acceptance control for M-CAC traffic is based on bandwidth available to the entire group. With this policy, it is possible that a video call in a "discount," adjustable rate class, say, is accepted by the edge router, even if the WFQ allocation is exceeded. If the M-CAC portion of the bandwidth becomes later congested, the performance of the discount video call will degrade. The source may react to packet loss by a reduction in data rate.

We have been carrying out experiments with aggregated class measurements. The initial results look promising. They confirm the importance of exploiting Q-OSPF routing, per class WFQ scheduling and multiple CAC styles in order to accommodate customers with different needs and strike the best balance between QoS guarantee and link capacity utilization.

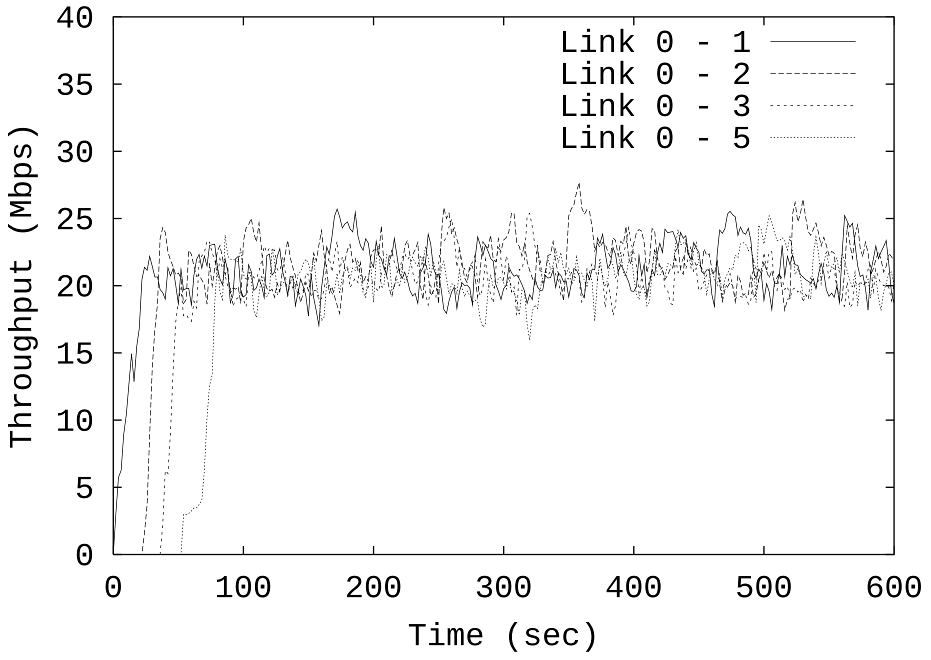


Fig. 7. Throughput of the queues on the bottleneck links when M-CAC is used.

5 Conclusions

In this paper, we have introduced three methods to provide QoS guarantees in DiffServ environment. They are all scalable, but strike a different tradeoff between implementation complexity, use of network resources and "hardness" of the QoS guarantees. They are all based on traffic shaping at border routers, eliminating costly, non-scalable re-shaping at core routers. Full scalability is achieved in core routers by operating on behavior aggregates, without addressing any single flow. The first two techniques that we tested (M-CAC and RA-CAC) yielded opposite results regarding performance. In fact, while measurement based CAC provides hard guarantees at the expenses of channel utilization efficiency, the measurement based CAC provides high utilization efficiency but frequent violations of delay bounds. This prompted the design of a midway solution, hybrid CAC, which is based on the same allocation rules as RA-CAC, but estimates allocation using measurements. The hybrid scheme increases somewhat the utilization efficiency without incurring in the performance degradation problems of measurement based CAC. Moreover, as the number of flows in the bottleneck grows large, the load fluctuations are reduced, and the delay bounds are easily respected by H-CAC. This may not true for a small number of bursty flows, where measurements may not provide reliable measurements.

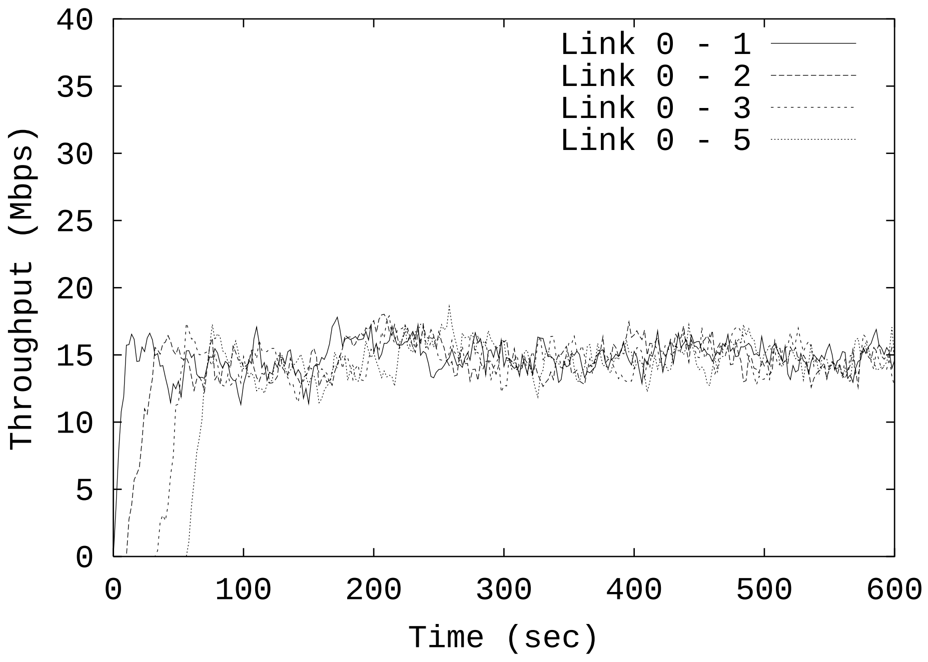


Fig. 8. Throughput of the queues on the bottleneck links when RA-CAC is used.

References

1. S. Blake et al., *An Architecture for Differentiated Services*, IETF RFC 2475, December 1998.
2. S. Shenker et al., *Specification of Guaranteed Quality of Service*, IETF RFC 2212, September 1997.
3. F. Le Faucheur et al., *MPLS Support of Differentiated Services*, Internet Draft , work in progress, March, 2000.
4. G. Apostolopoulos et al., *QoS Routing Mechanisms and OSPF Extensions*, IETF RFC 2676, August 1999.
5. M. R. Garey and D. S. Johnson, *Computers and Intractability*, Freeman, San Francisco, 1979.
6. R. Guerin, A. Orda, and D. Williams, *QoS Routing Mechanisms and OSPF Extensions*, Proceedings of IEEE GLOBECOM97, Vol. 3, pp. 1903-1908, Phoenix, Arizona.
7. C. Pornavalai, G. Chakraborty, and N. Shiratori, *QoS Based Routing Algorithm in Integrated Services Packet Networks*, Proceedings of International Conference on Network Protocols, Atlanta, Georgia, pp. 167-175.
8. W. Zhao and S. K. Tripathi, *Routing Guaranteed Quality of Service Connections in Integrated Services Packet Networks*, Proceedings of International Conference on Network Protocols, Atlanta, Georgia, pp. 175-182.

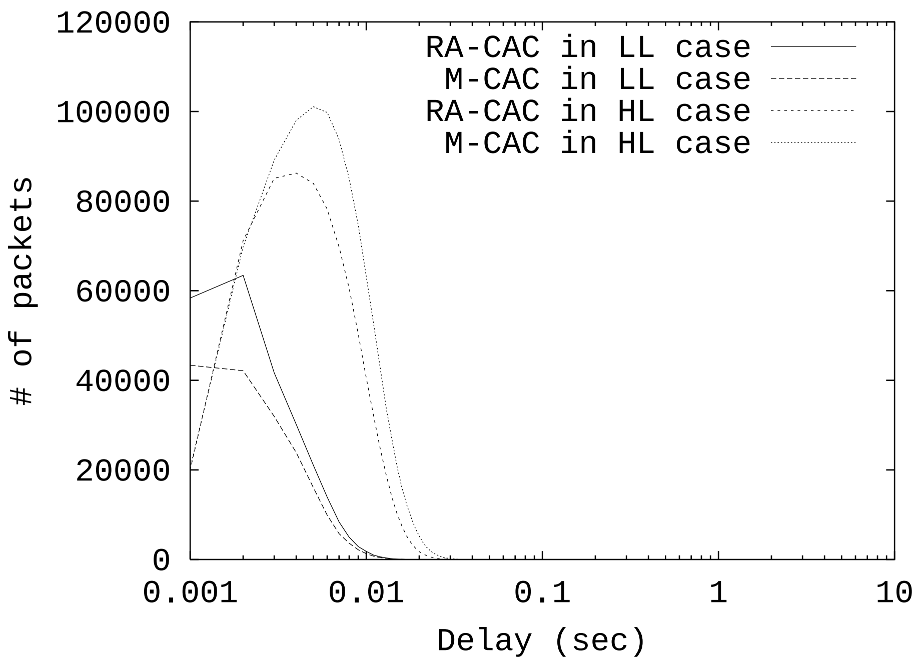


Fig. 9. End-to-end delay distributions when each CAC is deployed with different traffic load. (LL = lightly loaded, HL = highly loaded)

9. D. Cavendish and M. Gerla, *Internet QoS Routing using the Bellman-Ford Algorithm*, Proceedings of IFIP Conference on High Performance Networking, Austria, 1998.

10. S. Jamin, P. B. Danzig, S. Shenker, and L. Zhang, *A Measurement-based Admission Control Algorithm for Integrated Services Packet Network*, IEEE/ACM Transaction on Networking, 5(1): 56-70. Feb. 1997.

11. R. J. Gibbens and F. P. Kelly, *Measurement-based Connection Admission Control*, 15th International Teletraffic Congress, June 1997.

12. M. Grossglauser and D. Tse, *Measurement-based Call Admission Control: Analysis and Simulation*, Proceedings of IEEE INFOCOM'97, Kobe, Japan, April 1997.

13. C. Casetti, J. Kurose, D. Towsley, *An Adaptive Algorithm for Measurement-based Admission Control in Integrated Services Packet Networks*, Computer Communications, vol. 23(14-15): 1363-1376, August 2000.

14. E. Rosen, A. Viswanathan, and R. Callon, *Multiprotocol Label Switching Architecture*, Internet Draft, work in progress.

15. J. Heinane et al., *Assured Forwarding PHB Group*, IETF RFC 2597, June 1999.

16. V. Jacobson et al., *An Expedited Forwarding PHB*, IETF RFC 2598, June 1999.

17. E. W. Knightly and N. B. Shroff, *Admission Control for Statistical QoS: Theory and Practice*, IEEE Networks, March 1999.

Table 4. CAC statistics with number of admitted/rejected calls and relevant delay bound violation.

Lightly loaded case		
CAC scheme	M-CAC	RA-CAC
# of admitted calls	125	144
# of rejected calls	0	7
% of packets over 100-ms threshold	0 %	0 %
Highly loaded case		
CAC scheme	M-CAC	RA-CAC
# of admitted calls	397	291
# of rejected calls	822	877
% of packets over 100-ms threshold	0 %	0 %

18. R. Bagrodia, R. Meyer, M. Takai, Y. Chen, X. Zeng, J. Martin, B. Park, and H. Song, *Parsec: A Parallel Simulation Environment for Complex Systems*, Computer, Vol. 31(10), October 1998, pp. 77-85.
19. A. Dubrovsky, M. Gerla, S. S. Lee, and D. Cavendish, *Internet QoS Routing with IP Telephony and TCP Traffic*, IEEE ICC 2000, New Orleans, Louisiana, June 2000.
20. M. Gerla, C. Casetti, S. S. Lee, G. Reali, *Resource Allocation and Admission Control Styles in QoS DiffServ Networks*, Technical Report, UCLA, Los Angeles, USA, October 2000.

A Multicast Transport Service with Bandwidth Guarantees for Diff-Serv Networks[⋆]

Elena Pagani, Gian Paolo Rossi, and Dario Maggiorini

Dip. di Scienze dell'Informazione, Università degli Studi di Milano
{pagani, rossi, dario}@dsi.unimi.it

Abstract. The Int-Serv and Diff-Serv differentiation approaches, which have been proposed by the IETF, are in practice unable to provide QoS guarantees to the emerging multicast-enabled applications by means of a scalable mechanism able to support QoS services on a per-call basis and dynamic group memberships.
This paper describes a new approach that is capable of ensuring bandwidth guarantees to multicast sessions on IP-based networks and satisfies the above requirements. It includes a scalable, end-to-end *Call Admission Multicast Protocol* (CAMP) that operates as a sort of distributed bandwidth broker and allows to combine the benefits of both the IS and DS approaches in a single approach which is simple, scalable, operates on a per-call basis and supports the group membership dynamics.

Keywords: Quality-of-Service, multicast, call admission control, diff-serv domains, bandwidth broker, performance evaluation.

1 Introduction

A great deal of efforts has been recently devoted to provide through the Internet a transport platform capable of satisfying the Quality-of-Service (*QoS*) requirements of the emerging multimedia and real-time applications. This goal cannot be achieved without the introduction of some traffic differentiation policy, that changes the best-effort nature of the Internet protocols into a new generation of protocols providing the QoS guarantees (e.g., in terms of delay, bandwidth, jitter [7,2]), and addresses the following general application requirements:

multicast: most of the video, voice and multimedia applications require the support of one-to-many or many-to-many interaction schema [6];
scalability: the QoS-capable transport mechanism should scale to the entire Internet and to large multicast groups;
dynamic sessions: the QoS-capable transport mechanism should accept the dynamic opening/closing of QoS sessions on an on-demand basis;

[⋆] This work was supported by the MURST under Contract no.9809321920 "Transport of multicast packets with QoS guarantees".

group membership: the membership of the multicast group may be *static*, i.e. the result of an announcement and subscription process through, for instance, `sdr` [11], or *dynamic*. In the former case, the source can be aware of the group membership, while, in the latter, the source is generally unaware of both the membership and the cardinality of the group over time.

The Int-Serv (IS for short [5]) and Diff-Serv (DS for short [3]) differentiation approaches, which have been proposed by the IETF, do not completely satisfy the above requirements and, although they consider the multicast communications, in practice, they do not offer a satisfactory solution for managing group membership and session dynamics. IS can dynamically provide QoS guarantees on a per-flow basis through RSVP [18]. To do that, it requires substantial changes in the router architecture; each router maintains per-flow status information by running a hop-by-hop information exchange protocol. This makes RSVP hard to scale and unusable, in practice, when large multicast groups are considered. DS, on the contrary, considers flow aggregates and all packets belonging to the same aggregate receive the same network service on the base of the TOS byte in their IP header (DS byte in IPv6) [13], that identifies, and differentiates, the packet's *Per-Hop-Behaviour* (PHB). The DS model is simple and more scalable because it confines the QoS complexity at the *border routers* (i.e., the routers of the access network or the routers on a domain border) and leaves the core routers nearly unchanged. This approach requires that an end-to-end admission mechanism (the *bandwidth broker* [14]) dynamically controls and limits the amount of incoming flows to be assigned to a given aggregate. So far, such an end-to-end admission mechanism has not been designed for multicast-enabled sessions. As the consequence, the current QoS multicast experiences (e.g., QBone [10]) still consider a static environment where the amount of multicast QoS sessions and the group membership are both statically defined in a sort of permanent multi-point virtual circuit.

This paper describes a new approach that is capable of ensuring bandwidth guarantees to multicast sessions on IP-based networks and satisfies the above requirements. It includes a scalable, end-to-end *Call Admission Multicast Protocol* (CAMP) that allows to combine the benefits of both the IS and DS approaches in a single approach which is simple, scalable, operates on a per-call basis and supports the group membership dynamics. CAMP performs the functionalities of a distributed bandwidth broker and has been designed to operate within the RTP/RTCP protocol suite [16] and on top of a Diff-Serv network. When the membership dynamically changes, CAMP utilizes a CAMP *proxy* which is instantiated within the core, intree router involved in the topology change. The CAMP proxy however, autonomously and locally finds the needed information and stays active for the call set-up time only. Then, it turns off and the core routers have not to maintain status information.

As far as we know, this is the first attempt to address all the above requirements for multicast traffic in DS networks. A similar approach has been recently followed in [4] and applied to point-to-point video/voice sessions. The simulation results are encouraging and indicate that the approach is efficient for CBR traffic. Although the transition to support VBR traffic seems affordable, this will be part of the forthcoming research.

The paper is organized as follows: in Section 2, we describe the basic version of the algorithm, and we introduce the system architecture. In Section 3, we discuss the modifications needed to support dynamic changes in the multicast group membership. In Section 4, we present our simulation results.

2 Call Admission Multicast Protocol

2.1 The System Architecture

In this work we consider a network $\mathcal{N}$ of interconnected LANs. The network constitutes a *domain* [3], that is, a contiguous set of routers which operate with a common set of service models, service definitions and provisioning policies. Throughout this work we consider a Diff-Serv domain, in which the QoS is provided on a per-packet basis.

We assume that the routers in $\mathcal{N}$ are *multicast-enabled*, QoS-unaware, and that they are configured to use the *priority* packet scheduling policy. A membership protocol is used (e.g., IGMP [8]) to discover the group membership of the attached LANs. A multicast routing protocol (e.g., DVMRP [15]) maintains a tree-based multicast infrastructure among the routers, but Camp is independent of the underlying multicast protocol.

Camp is implemented as a module of the RTP library, which is linked to the application (figure 1). It performs the call set-up of an RTP session and is activated once the application calls RTP passing the traffic profile, through, for instance, a `TSpec` [17]. Camp exploits the services of RTP and RTCP [16]. RTCP is used to monitor the QoS provided at the recipients. RTP can be used at the receiver side to appropriately perform the playback of the source transmission. Upon the termination of the call set-up phase, the RTP session data transfer is initiated or terminated accordingly.

2.2 Camp with Static Membership

This section describes the Camp protocol that applies to static group membership. In this case, a session announcement protocol (e.g., `SDAP` [11]) can be used to announce the needed session information, such as the start time and the multicast address. Receivers perform an explicit join procedure and the group membership is defined before the transmission starts. The transmission source is aware of the membership of the group of recipients. Under these conditions, a multicast session is divided into two phases: the *session set-up* phase, performed by Camp, and the *data transfer* phase, performed through RTP. In the set-up phase, Camp uses the information supplied by the application to test if the network can guarantee the requested bandwidth. The test consists of sending `probe` packets to the destinations, so that the generated probing traffic has the same profile of the specified data traffic. The `probe`s are forwarded along the multicast

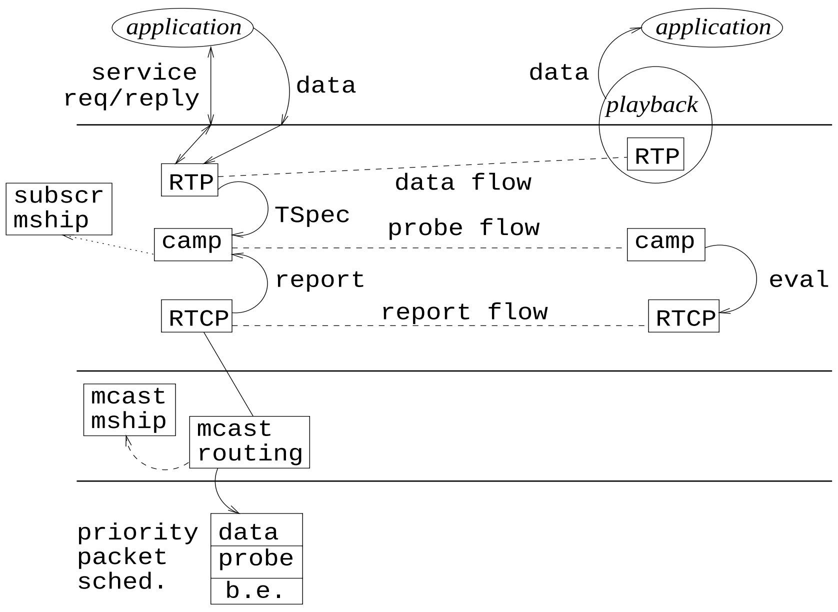


Fig. 1. Architecture of a Camp-based end station

routing tree. The basic idea consists of differentiating `probe`, QoS data and best effort packets by means of three priority levels. The `probe` packets are marked with a higher priority than the best effort packets and a lower priority than the QoS data packets. This priority assignment ensures that the probing traffic does not affect the existing QoS flows. On the other hand, `probe` packets can drain the available bandwidth for new QoS flows at the expenses of the best effort traffic.

At the receiver side, the Camp entity evaluates the quality of the received probing traffic. The source diffuses its own traffic profile by using the RTCP reports. Each receiver r matches that profile against the received probing traffic profile and decides whether to accept the whole transmission or not. r informs the source about its decision with its own RTCP reports. A receiver that refuses the service unsubscribes from the group. As the consequence, the corresponding router may be pruned from the multicast routing tree.

If the service is accepted by at least one recipient, then the source switches from the transmission of `probe` packets to the transmission of the data packets generated by the source application, without discontinuity. Data packets are forwarded along the pruned tree. The priority policy ensures a sort of resource reservation: it guarantees that the bandwidth that has been 'assigned' to a given flow during its probing phase cannot be later assigned to other `probe` or best effort packets. Moreover, priority naturally provides flow aggregation.

The knowledge of the group membership is needed because the source *must* wait for the reports from *all* the receivers, before switching to the data transmission. The example of figure 2(a) explaines this point. The source generates a QoS flow $flow_1$. If it receives positive reports from a subset of receivers, let us say those in the subtree A, it knows that some destination is available for the data flow. If the source switches to the data transmission before receiving the reports from the receivers in the subtree B, other established QoS flows could be negatively affected. In fact, let us consider the flow $flow_2$ that traverses the B's branches, and does not leave enough bandwidth for $flow_1$. As the consequence, the receivers in B would eventually send negative reports and prune to refuse $flow_1$. If the switch is performed before receiving all the reports (i.e., before the needed prune operations), the data packets of the aggregate flowing on the subtree B will be dropped, independently of the flow they belong to.

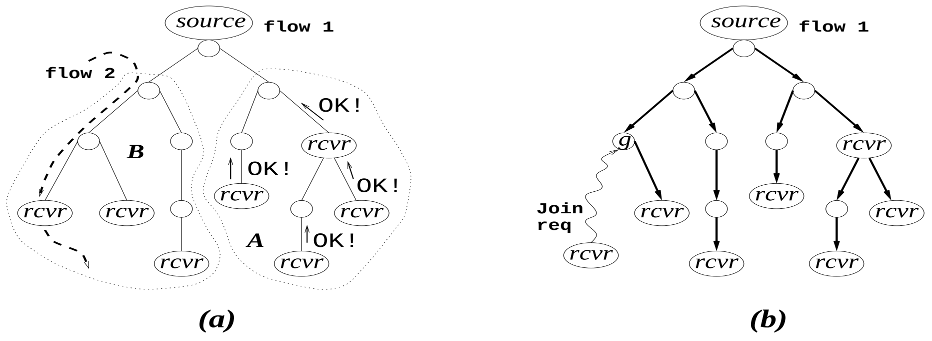


Fig. 2. (a) Example of concurrent flows. (b) Example of dynamic group membership

The need of obtaining all the decisions requires that report loss is prevented, for instance by having RTCP that uses the TCP services. The source can notice when all the reports have been received, because, by hypothesis, it knows the (static) destination group membership beforehand. The case of dynamic group membership is discussed in the next section.

3 Supporting Dynamic Group Membership

In several applications, such as news services and video broadcasting, the source does not need, or desire, to be aware of the receivers group membership, while receivers are free to join and leave the group dynamically. In this new model, the source announces the multicast session via SDAP and starts transmitting at the scheduled time if at least one receiver is listening. In all these cases, the Camp model described above does not apply, because the tree infrastructure that supports the session flow is unable to maintain the bandwidth guarantees in case of topology changes. To understand where the problem arises, let us consider the example shown in figure 2(b), where a receiver wants to join the group when the transmission of a flow $flow_1$ is established. According to the multicast routing protocol, the joining node would graft to an intree router, say g. Anyway, the data packets cannot be forwarded along the new branch, without testing if enough resources are available to accommodate them, for the same arguments discussed before (see figure $2(a)$). The probing phase cannot be started at the root, to avoid the duplication of the resource reservation already installed for $flow_1$ on the path from the root to g. On the other hand, the probing can neither be started by g, which is QoS-unaware. In fact, $flow_1$ shall be transported through the new branch as part of the proper PHB to which it belongs and, in line with the DS model, the involved intree router should be unable to identify the subset of packets belonging to $flow_1$.

3.1 The Camp Proxy

To overcome the above limitations we decided to install a Camp *proxy* in the in-tree routers involved in the membership change. The proxies are in charge of managing the joining nodes.

When a new node wants to join a QoS multicast session, it explicitly joins the proper group of receivers (through, for instance, IGMP report exchange). Let us indicate with f the transmission flow and with G_f the group of the receivers of f. As soon as an intree router g, belonging to the tree T_f for G_f, creates a new downstream interface for T_f, it instantiates the Camp proxy, say p_f^g, whose lifetime lasts until the set-up phase for the new host terminates. Once activated, the proxy receives the identifier of the new interface and generates, by gathering the information from the local routing table, the data structure of figure 3, where it temporarily maintains the status of the active probing phases. If a new join request is received while the proxy is active, a new entry of the data structure is created. The proxy turns off when all the entries in the data structure have been deleted (service refused) or switched to data transfer.

probing interfaces	*state*
output interface 1	data
...	...
...	...
...	...
output interface n	probing

Fig. 3. Data structure maintained by a proxy

During its activity the proxy re-marks as `probe` packets all the incoming QoS data packets that should be routed to the probing interfaces. Moreover, the proxy snoops all the packets received from the downstream interfaces. When a RTCP report is received, p_f^g temporarily records it and drops it to prevent its forwarding. Should the report be positive, the proxy changes the corresponding entry from the state *probing* to *data*. Otherwise, it removes the entry. This simple mechanism solves the problem due to a dynamically changing tree topology and achieves the following goals:

1. the "`probe`" packets allow to evaluate the resource availability only along the new branch;
2. the new destination immediately starts receiving the real data flow, although possibly with a lower quality than required;
3. the data sent to the new destination do not affect previously established flows traversing the new branch, as the data packets are marked and treated as `probe` packets;
4. the membership is hidden to the source.

To show that the proxy mechanism is in practice very general, let us consider the system condition when the multicast session is initiating. At that time, we assume that a certain amount of recipients joined the multicast tree (in the trivial case of no recipients the transmission would not be initiated to avoid the waste of bandwidth) and that the local CAMP proxies are active as described before waiting for some packet to forward downstream. At the scheduled session time, the CAMP protocol at the source host will start the probing phase. CAMP proxies act as described: remark (although uselessly) the incoming `probe` packets into outgoing `probe` packets and wait for snooping RTCP reports. In addition, they understand they are running a probing phase because of the type of the incoming packets. As long as they receive `probe` packets, they snoop RTCP reports to update the local data structure, but forward them upstream. When the source CAMP entity receives at least one positive report from the group, it switches to the transmission of the data and both the end systems and the proxies operate as described.

4 Performance Evaluation

We have implemented the architecture shown in figure 1, in the frame of the NS-2 simulation package [9], to evaluate how CAMP behaves when several QoS flows compete with the best effort traffic for the available bandwidth. We considered the DVMRP multicast routing protocol and we embedded a real RTP implementation [1] into the RTP template of NS-2. We simulated a meshed network of 64 nodes, connected by optical links of 2 Mbps bandwidth and variable length in the range 50 to 100 Km. The size of `probe`, best effort and data packets is 512 bytes. The background best effort traffic is provided by 32 unicast connections between randomly chosen source-destination pairs. Each best effort source generates traffic with a rate of 0.66 Mbps.

The recipients compare the received rate with the source rate: if the difference is below a tolerated threshold, a recipient sends a positive report. We performed measures for different thresholds. The decision is sent within the first RTCP report a destination generates after the reception of a number of `probe` packets, i.e. of samples, sufficient to ensure an accurate measure of the available bandwidth by covering the rate of the slowest traffic source. In all our simulations, the recipients decide after the reception of 500 `probe` packets. This way, the length of the set-up phase depends on the source rate.

A first set of experiments has been performed with two CBR sources located on the same node, and the acceptance threshold at the receivers set to the 5% of the source rate. The first source generates traffic at the rate of 1 Mbps; the rate of the second source assumes different values in the range 0.4 Mbps to 1.9 Mbps. The probing phase of the second source starts 1.5 sec. after the start of the first source, so that the two set-up phases partially overlap. By performing experiments with different group cardinalities for the two groups we observed that the performance are almost independent of the group cardinality. In the following, we only report the results we obtained when both groups have

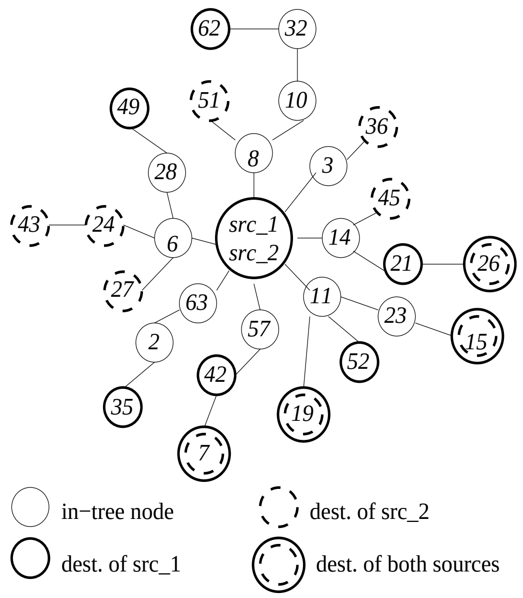


Fig. 4. Overlapped multicast trees for the QoS sources

10 destinations. Groups partially overlap; the simulated topology restricted to the multicast routing trees for the two groups is shown in figure 4. The group membership is static throughout the simulation.

Simulations indicate that CAMP effectively performs the call admission control. In fact, until the sum of the source rates is lower than the link capacity, the recipients in both groups accept the transmissions and receive at the correct data rate. When the rate of the second source is higher than 1 Mbps, the source transmission can be accepted only by receivers linked through separate branches to the multicast trees. When the multicast trees of the two groups share the same link, only the destinations in one group shall be able to accept the transmission. Camp correctly identifies the receivers able to receive at the proper data rate, removes the others from the group, and performs this task with the required fairness. In fact, it ensures that the transmission that started the probing phase earlier would not be negatively affected by the second group, and, when in competition, would receive the bandwidth guarantees as shown in figure 5. In figure 5(a) we report the number of recipients of both groups that accepted the transmission, with respect to the traffic generated by the second source. In figure 5(b) we show with a solid line the rate received at the destinations that accept the service. The dashed line represents the average rate obtained during the set-up phase by considering all the receivers, and also represents the data rate that would be observed at the destinations without the call admission control. As shown in figure 5(a), by setting the threshold to 2%, 5%, 10% of the source

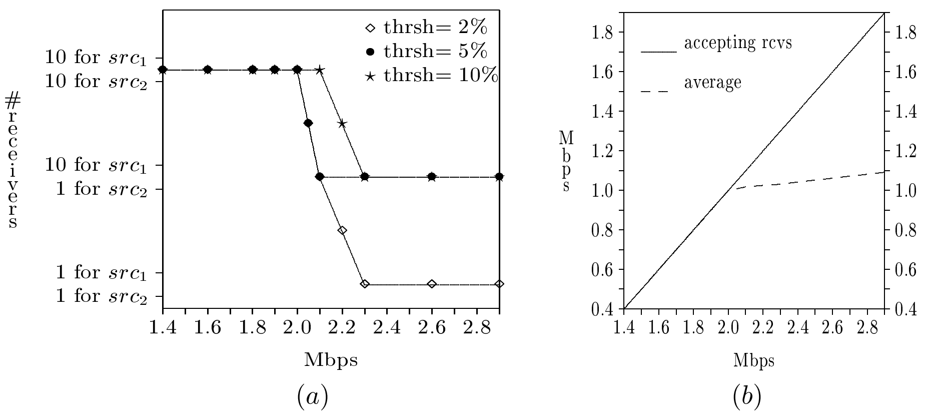
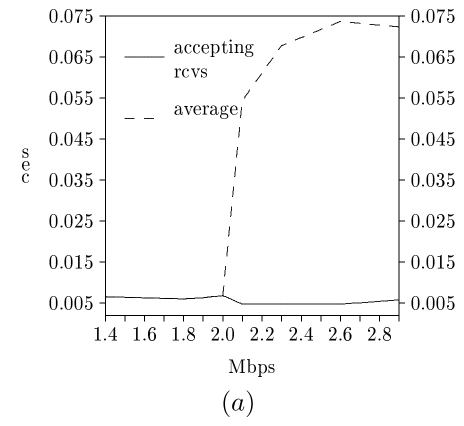


Fig. 5. (a) Overall number of accepting recipients for both groups, for different threshold values and $|G| = 10$. (b) Average received rate for the second group of recipients vs. offered load, for $|G| = 10$ and threshold value 5%

rate, Camp is able to properly provide the bandwidth guarantees to all the receivers in both groups until the data rate does not saturate the link capacity. At a higher request of bandwidth, and threshold at 5% and 10%, all the receivers in the first group accept the transmission while, in the second group, only the destinations on separate branches accept the transmission (in the simulation, only 1 receiver). As confirmed by the results of figure 5(b), the receivers able to accept the transmission receive the adequate bandwidth guarantees to receive at the correct data rate. When the threshold is set too low, 2%, the number of receivers able to accept the transmission becomes very low due to the observed delay and jitter, as discussed below. The acceptance threshold should be chosen according to the semantics of the source application, that is, it should be no greater than the maximum packet loss probability that can be tolerated so that the information does not become useless.

In figure 6, we report the observed end-to-end delay (figure 6(a)) and jitter (figure 6(b)). The jitter is the variance on the packet inter-arrival time and has been computed according to the algorithm given in the RTP specification [16].



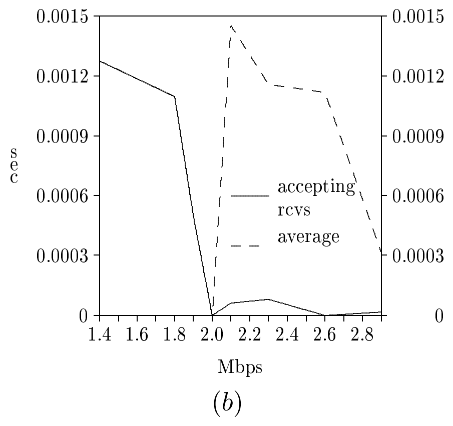


Fig. 6. (a) Average end-to-end delay and (b) jitter for the second group of recipients vs. offered load, for $|G| = 10$ and threshold value 5%

The end-to-end delay is almost constant before saturating the links, and it remains unchanged for the accepting destinations with congested links. This indicates that the flows characteristics are preserved from source to destination, independently of the network load and the group cardinality. The measure of the jitter shows the effects of the presence of best effort packets at the core routers. As expected, the priority mechanism alone is not sufficient to ensure delay and jitter control at the destinations. As the consequence, the jitter decreases together with the decreasing of the amount of best effort packets competing to use the link resources. Jitter approaches 0 when both the QoS sources generate at the rate of 1 Mbps (in this case, there is no space left for best effort traffic). At higher QoS rate some destination has to refuse the service (see figure 5(a)), thus allowing that a few best effort packets still succeed in accessing the network. As shown by figure 6(b), their effect is marginal over the links where the accepted traffic is forwarded.

By exploiting the RTP service is possible to annihilate the jitter, by delivering the received packets at a constant rate.

5 Concluding Remarks

In this paper we propose the Camp library to perform call admission control in diff-serv domains for multicast applications. Camp adopts an innovative approach, intermediate between the IS and the DS model, to support dynamic changes in the group of recipients. Camp proxies are dynamically created in the routers that must take care of those changes. They perform the updates in the system configuration required by the membership changes, and then they are destroyed. We implemented the described mechanism in the frame of the ns-2 simulation package. The achieved results are promising: the devised approach effectively performs admission control. The accepted services have the requested QoS guaranteed. Yet, further investigation has to be carried out concerning the interactions amongst several concurrent transmissions and their impact on the probability of successful service establishment.

So far, proxies can support dynamic multicast membership under the assumption that all the recipients grafted to the tree receive the same transmissions. We are extending their capabilities so that heterogeneous destinations, willing to accept different sets of data flows, can be supported. Similarly, the mixing and translation functionalities of RTP could be embedded into the proxies. In the near future, we will be able to perform experiments with the described architecture in the frame of a testbed that we are currently deploying over the departmental network [12].

References

1. `http://www.cs.columbia.edu/~hgs/rtp/`
2. Bagnall, P., Briscoe, R., Poppitt, A.: Taxonomy of Communication Requirements for Large-scale Multicast Applications. Internet Draft, draft-ietf-lsma-requirements-02.txt. Work in progress (1998)
3. Blake, S., Black, D., Carlson, M., Davies, E., Wang, Z., Weiss, W.: An Architecture for Differentiated Services. Internet Draft, draft-ietf-diffserv-arch-02.txt. Work in progress (1998)
4. Borgonovo, F., Capone, A., Fratta, L., Marchese, M., Petrioli, C.: PCP: A Bandwidth Guaranteed Transport Service for IP networks. Proc. IEEE ICC'99 (1999)
5. Braden, R., Clark, D., Shenker, S.: Integrated Services in the Internet Architecture: an Overview. RFC 1633. Work in progress (1994)
6. Bradner, S.: Internet Protocol Multicast Problem Statement. Internet Draft, draft-bradner-multicast-problem-00.txt. Work in progress (1997)
7. Bradner, S.: Internet Protocol Quality of Service Problem Statement. Internet Draft, draft-bradner-qos-problem-00.txt. Work in progress (1997)
8. Cain, B., Deering, S., Thyagarajan, A.: Internet Group Management Protocol, Version 3. Internet Draft, draft-ietf-idmr-igmp-v3-01.txt. Work in progress (1999)
9. Fall, K., Varadhan, K.: ns Notes and Documentation. The VINT Project: http://www-mash.CS.Berkeley.EDU/ns/ (1999)
10. Internet2 QoS Working Group Draft: Draft QBone Architecture. Internet Draft, draft-i2-qbone-arch-03.txt. Work in progress (1999)
11. Handley, M.: The sdr Session Directory: An Mbone Conference Scheduling and Booking System. http://mice.ed.ac.uk/mice/archive/sdr.html (1996)
12. Maggiorini, D., Pagani, E., Rossi, G.P.: A Testbed Environment for the Performance Evaluation of Modular Network Architectures. Proc. 1st IEEE European Conference on Universal Multiservice Networks (ECUMN 2000) (2000) 283–292

13. Nichols, K., Blake, S., Baker, F., Black, D.: Definition of the Differentiated Services Field (DS Field) in the IPv4 and IPv6 Headers. Internet Draft, draft-ietf-diffserv-header-04.txt. Work in progress (1998)
14. Nichols, K., Jacobson, V., Zhang, L.: A Two-bit Differentiated Services Architecture for the Internet. Internet Draft, draft-nichols-diff-svc-arch-00.txt. Work in progress (1997)
15. Pusateri, T.: Distance Vector Multicast Routing Protocol. Internet Draft, draft-ietf-idmr-dvmrp-v3-09.txt. Work in progress (1999)
16. Schulzrinne, H., Casner, S., Frederick, R., Jacobson, V.: RTP: A Transport Protocol for Real-Time Applications. RFC 1889. Work in progress (1996)
17. Shenker, S., Wroclawski, J.: General Characterization Parameters for Integrated Service Network Element. RFC 2215. Work in progress (1997)
18. Zhang, L., Braden, R., Berson, S., Herzog, S., Jamin, S.: Resource reSerVation Protocol (RSVP) - Version 1 Functional Specification. RFC 2205. Work in progress (1997)

Modeling the Stationary Behavior of TCP Reno Connections*

Claudio Casetti and Michela Meo

Dipartimento di Elettronica, Politecnico di Torino, Torino 10129, Italy,
{casetti,michela}@polito.it,
http://www1.tlc.polito.it/

Abstract. In this paper, we outline a methodology that can be applied to model the behavior of TCP Reno flows. The proposed methodology stems from a Markovian model of a single TCP source, and eventually considers the superposition and interaction of several such sources using standard queueing analysis techniques. Our approach allows the evaluation of such performance indices as throughput, queueing delay and segment loss of TCP flows. The results obtained through our model are validated by means of simulation, under different traffic settings.

1 Introduction

The TCP protocol plays a key role in delivering a reliable service to the most common network applications such as email programs and Web browsers. Over the years, several versions of the TCP protocol have been proposed, with the aim of enhancing its response to network congestion and data loss. However, the resulting complexity of the algorithms governing TCP's flow control dynamics has so far defied any accurate analytic representation.

In recent years, several efforts have been devoted to modeling such a complex protocol as TCP. Some papers have tried to capture the essential TCP dynamics through closed-form expressions. Lakshman and Madhow [6] and Kumar [5] use Markovian analysis to develop a closed-form expression for the throughput of TCP connections by observing the cyclical evolution of the TCP transmission window; the latter work introduces some extensions for several versions of TCP, incorporating such features as coarse timers, fast retransmit and fast recovery. A fluid model of a single TCP connection sharing a link with background, non-TCP traffic was proposed by Altman et al. [1], in a hybrid approach combining both analytical and experimental work; as was done in other, more recent studies, their model proceeds through a stochastic analysis of the dynamic behavior of TCP's congestion window size: specifically, they aim at providing an expression for the throughput of a single connection. Similarly, Mathis et al. [7] focused on the stochastic behavior of the Congestion Avoidance mechanism, deriving an expression for the throughput that was then applied to study the behavior of several flavors of TCP sources and queueing techniques. Recently, Padhye et

* This work was supported by the Italian Ministry for University and Research

al. [8] have developed a steady-state model that approximates the throughput of bulk TCP flows as a function of loss rate and round trip time, comparing their estimates with real-life traces of TCP traffic.

Notably, some researchers [9], [10] have chosen a different approach whereby the observation of "actual" TCP traces was the foundation of empirical models. These efforts imply collecting hours' (sometimes days') worth of data, and finding suitable statistical distributions for the observed data.

The methodology presented in this paper was first explored in [2], where only a simplified Tahoe version of the TCP protocol was studied. In the present work, we extend the model to the Reno version, including the Faster Recovery/Faster Retransmit mechanisms. Using our approach, key parameters of TCP traffic can be estimated, such as throughput, goodput, queueing delay and segment loss. The approach combines a Markovian model of a single TCP source in isolation, and the analysis of superposition and interaction of several TCP sources through standard queueing analysis techniques. We restrict our analysis to a simple bottleneck topology with several concurrent one-way TCP flows.

Specifically, our efforts are focused on mimicking common application-level behaviors, such as the On-Off activity of a Web server responding to clients' requests.

A validation of our model is provided through extensive comparisons of the model's results with the output of the LBL simulator, "*ns* version 2". Comparisons show that our model succeeds in providing an accurate representation of the behavior of TCP traffic under several different settings.

The paper is organized as follows: Section 2 provides a broad outline of the approach we have followed; Section 3 describes the model of the upper layers (application and TCP layers) and of the lower layers (the network). In Section 4 we validate our model by comparison of the analytical results against simulation results. Finally, conclusions follow in Section 5.

2 The Methodology: Reciprocal Model Tuning

A realistic representation of TCP traffic should take into account far too many components: the characteristics of the network (i.e. topology, routing, queueing capabilities at intermediate nodes); the details of the TCP protocol; the behavior of user applications.

We thus resort to approximate techniques, simplifying some aspects of the system while keeping a detailed description of the characteristics which we guess have a major impact on the system performance.

Basically, we divide the system into two parts: the TCP sources and the network; we then develop an analytical model for each part, assuming that each source can be considered statistically independent from the others. Although the two models are separately analyzed, we have to describe the interaction between them.

The first model uses a Markovian representation to describe the TCP source through the interaction between application-level behavior and transport-layer protocol dynamics. Complex scenarios featuring several such sources require an aggregation of these models (based on the same principles but with different values for the parameters), each corresponding to a different class of TCP connections. Classes may differ in the TCP set-ups, in the characteristics of the application, or even in the propagation delay, or in the routing path. Figure 1 sketches the proposed approach for a system with n different classes of TCP connections.

The second model describes the network as a queueing network, bringing into the picture aspects such as topology, queueing capacity at the nodes, link capacity.

Let us first focus on the effect that sources have on the network. The TCP sources act on the network by injecting PDUs into it. Following TCP terminology, we will refer to PDUs as "segments". The segment generation process is rather complex and depends on the status of all sources. However, we simplify it by deriving the average traffic Γ_i, which a TCP source of class i generates. The traffic that the whole set of class i sources generates is given by $N_i\Gamma_i$ where N_i is the number of sources belonging to class i. The aggregate traffic λ can therefore be computed by summing the contribution from all n classes as $\lambda = \sum_{i=1}^{n} N_i\Gamma_i$. We are assuming that the segment generation process as seen by the network is Poisson with parameter equal to λ. Of course, this assumption introduces an approximation: we lose the correlation imposed by the network on the sources behavior.

The network influences the behavior of the sources by means of the feedback information inherent to TCP, so as to let the source rate adapt to the current workload of the network. Sources change their rate when segment losses are detected, and on the basis of their estimates of the delay perceived by outgoing segments. From the network model, we derive indications of the average segment loss probability and delays and we import these indications into the source model. The approximation here consists in using average measures instead of complete distributions for losses and delay; furthermore we decouple the states of the network and of the sources.

The stationary behavior of the system is derived through a sequence of successive refinement steps in which the two models tune each other.

From the steady-state solution of the Markov chain modeling the TCP source, we derive λ, the average aggregate traffic entering the network. The input traffic λ is fed to the network model and both segment loss probability and queueing delay are derived; in their turn, these quantities are used to refine the TCP source model. Once these values are tuned, we repeat the analysis of the source model and get a new value for λ, which is used to further refine the network model; and so on.

This procedure of reciprocal tuning of the two models is repeated until further analysis steps produce negligible adjustments in the value of the parameters. At that point, the model can be said to provide predictions of the stationary behavior of the system.

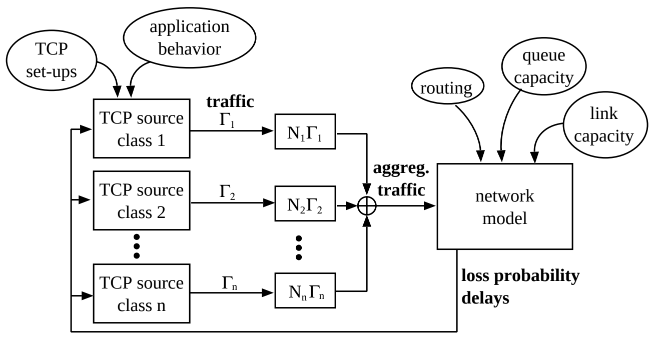


Fig. 1. TCP methodology

In Section 3.1 and Section 3.2 we describe the TCP source model and the network model, respectively. For the sake of readability, the description focuses on the specific case of a single class of TCP connections, and a simple network with a single bottleneck.

3 Model Description

In this Section, we present a model for sources exhibiting an ON/OFF behavior at the application layer, while employing a TCP Reno connection to send data. A TCP connection resets its parameters (i.e., window size, slow start threshold, etc.) at the beginning of every ON cycle. While we acknowledge that this is not necessarily a realistic behavior under all circumstances, we believe that, in spirit, it may be used to represent web-like transactions.

3.1 Model of the Source

The traffic injected into the network is generated by a rather complex interaction between the TCP protocol at the transport level and the application using the TCP connection. If we are to model this complex behavior, we necessarily have to take into account aspects of both layers.

We develop a hierarchical model of the source, where hierarchical levels correspond to protocol layers. We will start by describing the model at the application level, then we will look at the TCP level, and, eventually at the "compound" model as the aggregation of both application- and TCP-level behaviors.

Application Level The application is assumed to have an ON/OFF behavior; thus, only two states are possible: *idle*, I for short, and *active*, or A. When in idle state the application has no active TCP connection. On the contrary, when in active state, the application is sending data over a TCP connection.

We assume that the time spent in each state is a random variable with negative exponential distribution; letting T_{OFF} and T_{ON} be the average time spent respectively in state I and A, the parameters of the negative exponential distribution are $\alpha = 1/T_{OFF}$ and $\beta = 1/T_{OFF}$. This behavior can be modeled by a two-state Markov chain.

At a lower hierarchical level, it is possible to detail the model that takes into account the presence of the TCP protocol, focusing on state A (no TCP connections are active in I).

TCP Level The Reno version of TCP, as described in [11], essentially operates in four states: Slow Start, Congestion Avoidance, Fast Retransmit and Fast Recovery. The core of the protocol relies on the *congestion window*, whose size w determines the maximum number of segments a source is allowed to send pending acknowledgement. The window size can grow up to a maximum value W_M, its increments and decrements being aimed at making the protocol adapt to the operating conditions of the network. Often, the protocol dynamics are affected by the buffering and processing capabilities at the receiver, through the receiver window. However, in our analysis, we assumed that the receiver window never interferes with the congestion window (i.e., it is assumed that it is equal to W_M and that received segments are processed at once). Although TCP represents the window size with the granularity of bytes, it is common practice to translate the size using the Maximum Segment Size (MSS) as a unit. We will follow such notation from now on.

Right at the onset, w is set equal to 1 and grows according to the Slow Start mechanism: every ACK received by the transmitter widens the congestion window by 1, providing buffer space for the transmission of two more segments. As a result, the congestion window size w doubles roughly every round trip time, thus exhibiting an exponential behavior. Some implementations of TCP employ the so-called *delayed ACK* technique, which provides for sending a cumulative ACK every two segments. In our model we did not take into account such feature, although, as will be clear, its introduction would require marginal modifications to the model itself. The Slow Start phase ends when the window size reaches a threshold, W_t (we assume that, at connection startup, $W_t = W_M/2$).

When the window size w grows larger than the threshold W_t, the Congestion Avoidance state is entered. When in this state, every received ACK triggers an increment of the congestion window by $1/w$-th its own size. The window increase algorithm in the Congestion Avoidance state is roughly equivalent to enlarging the window size by one unity when all ACKs for the segments sent in the previous window are received. Thus, in practice, the window size increases linearly until it reaches its maximum allowed value W_M.

A segment loss is detected either by a timeout mechanism or by three consecutive duplicate ACKs. Whenever a segment is transmitted, a timeout starts; if no ACK is received before the timeout expiration, the segment is considered lost. In this case, the threshold is set to half the current window size ($W_t = w/2$), the window is reset to one segment, and the Slow Start phase is entered by re-transmitting the first unacknowledged segment.

To speed up the loss detection process and avoid excessive shrinking of the congestion window, duplicate acknowledgements are taken into account. Since the TCP receiver is supposed to send a duplicate ACK upon the reception of an out-of-sequence segment, after a lost segment several duplicate ACKs are usually generated, depending on the window size when the loss occurs. The reception of three duplicate ACKs is therefore a strong indication that a segment was lost. The TCP source thus retransmits the missing segment (Fast Retransmit) and enters a state called Fast Recovery, during which the threshold is set to half the current window size and the window size itself is subsequently shrunk to the threshold plus three segments. As a result, the missing segment is retransmitted more promptly than after a timeout, and the Slow Start phase is not entered [1].

Summarizing, in order to derive a Markov chain model of the dynamics of a TCP connection the following details have to be accounted for: i) the congestion window size, ii) threshold changes, iii) a segment loss indication, and iv) the protocol state (Slow Start, Congestion Avoidance, or Fast Recovery/Retransmit).

To reflect these observations, the states of the Markov chain describing a TCP connection were represented as vectors with three variables $s = (w, W_t, l)$, where w denotes the current window size, W_t is the current value of the threshold, l is an indication that either no loss has occurred ($l = 0$) or that a loss occurred but was not yet recovered ($l = 1$). The protocol state will be accounted for by an ad-hoc selection of the Markov chain states. Below, we further discuss our modeling choices regarding these key metrics.

- The congestion window size w is measured in segments, as we stated earlier; in principle, w may take all integer values between 1 and W_M. This choice results in a time-granularity of the order of the round-trip time for the dynamics of the window size increments.
- Consistently with w, W_t too is measured in segments: only those values of W_t that the window may take in the Slow Start exponential growth are accounted for. For example, consider the case of W_M equal to 20. W_t is initially set to 10 and the window size grows in Slow Start taking the values: 1, 2, 4, 8. Then, after a fraction of the round trip time, the window size reaches the value 10 and enters Congestion Avoidance. However, a time-granularity of the order of the round trip time does not allow us to distinguish this situation from the one where W_t is equal to 8, therefore we *approximate* the actual behavior by having the protocol change its state (and the window start growing linearly) when $w = 8$ rather than when $w = 10$. As a consequence, Slow Start is modeled only for a handful of threshold values and, specifically, *all powers of 2* between 1 and $W_M/2$. In the case mentioned above ($W_M = 20$) the only possible values for W_t are 1, 2, 4, 8.
- The model uses the boolean indicator l to identify those periods in which a loss occurred, but either the timeout has not yet expired, and the loss is still

[1] For reasons of space, a more detailed description of the Fast Recovery/Fast Retransmit is left out, and can be found in [11]

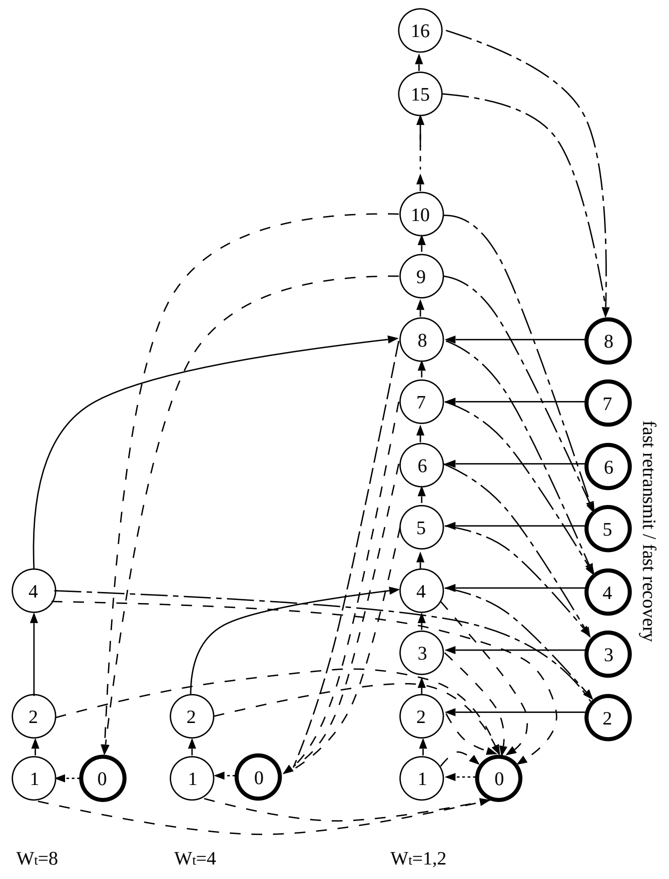


Fig. 2. TCP window size increment and decrement, for W_M =16

undetected, or a triple duplicate ACK was received and the source is in Fast Recovery state. We introduce this variable, which does not have an actual correspondence with the TCP protocol, in order to distinguish the periods in which the window dynamics are fast, from the periods in which TCP slows down due to timeout or duplicate acknowledgements. Therefore, $l = 0$ denotes periods with no loss, while $l = 1$ denotes periods with "unrecovered" losses.

Figure 2 shows the state diagram of the Markov chain describing the TCP window dynamics when $W_M = 16$. States represented by thin-edged circles correspond to $l = 0$ (no losses), while thick-edged circles represent states where a loss is not yet recovered ($l = 1$). We omitted l and W_t in the label of the states to make the figure more readable. In the $l = 0$ section, we recognize 3 branches of the chain: from left to right, each represents the Slow Start phase with W_t

equal to 8, 4 and 2 (or 1, since the two yield the same behavior); incidentally, the $W_t = 2$ branch can also be used to represent a congestion-avoidance growth. States with $l = 1$ represent either the Fast Recovery phase (rightmost branch) or the wait for a timeout expiration. The solid lines indicate transitions of the window size increment, which can either be linear (during Congestion Avoidance) or exponential (during Slow Start). The transition rate is computed as the inverse of the average time before a window increment, (let it be denoted as δ), weighted by the probability of all segments being correctly delivered, that we presently indicate as P_{noloss} (we will discuss segment loss probability later on). Therefore, if we indicate the round trip time as θ, we can safely assume that if no losses occur, the next window increment occurs after a round trip time, hence $\delta = 1/\theta$. The round trip time θ is given by two components: the queueing delay Q and the two-way propagation delay Q_P; thus we can derive δ from:

$$\frac{1}{\delta} = \theta = Q + Q_P \tag{1}$$

While Q_P is a static measure of the physical distance between transmitter and receiver, the queueing delay Q depends on the congestion of the network.

The line-point dashed lines are used to represent transitions toward Fast Recovery states. The dashed lines represent transitions toward timeout states (marked '0') [2], when no segments are sent. It should be noted that window sizes smaller than 4 never allow the source to exploit Fast Retransmit/Recovery when it is in Congestion Avoidance: indeed, the number of duplicate ACKs is never larger than 2, one ACK short of triggering Fast Retransmit. Window sizes between 4 and 10 lead to Fast Retransmit/Recovery if only one segment (from a window of data) is lost, while they cause a timeout expiration if more than one segment is lost (from the same window of data). Fast Retransmit/Recovery for window sizes larger than 10 is entered both for single and double losses, while three losses from the same window of data force the sender to wait for a retransmission timeout whenever the number of segments between the first and second dropped segments is less than $2 + 3W/4$, where W is the congestion window just before the Fast Retransmit. A more detailed analysis of these cases can be found in [3].

Following the above observations, we are presented with the following cases:

- When one or more losses occur with congestion window $w > 10$, the chain moves from states $s = (w, W_t, 0)$ with $w \in [11, W_M]$ to states $s = (w_{fr}, W_t, 1)$ with $w_{fr}, W_t \in [2, \lceil W_M/2 \rceil]$ and $w_{fr} = W_t = \lceil w/2 \rceil$ (Fast Retransmit/ Recovery). The transition rate is equal to $\delta \cdot (1 - P_{noloss})$. Clearly, we consider as negligible the probability that three or more losses lead to a timeout expiration.
- When a single loss occurs with congestion window $4 \leq w \leq 10$, the chain moves from states $s = (w, W_t, 0)$ with $w \in [4, 10]$ to states $s = (w_{fr}, W_t, 1)$

[2] although in actual TCP implementations the window size never becomes 0, in our model we chose $w = 0$ to stress the fact that no segments are transmitted while waiting for the timeout to expire

with $w_{fr}, W_t \in [2,5]$ and $w_{fr} = W_t = \lceil w/2 \rceil$ (Fast Retransmit/Recovery). The transition rate is equal to $\delta \cdot P_{1loss}$, where P_{1loss} denotes the probability of losing only one segment.

– When two or more losses occur with congestion window $4 \leq w \leq 10$, the chain moves from states $s = (w, W_t, 0)$ with $w \in [4, 10]$ to states $s = (0, W_t, 1)$ with W_t chosen as the power of 2 closest to $w/2$ (Timeout). The transition rate is equal to $\delta \cdot (1 - P_{noloss} - P_{1loss})$.
– When any number of losses occur with congestion window $w < 4$, the chain moves from states $s = (w, W_t, 0)$ with $w \in [1, 3]$ to states $s = (0, W_t, 1)$ with W_t chosen as the power of 2 closest to $w/2$. The transition rate is equal to $\delta \cdot (1 - P_{noloss})$.

It should be noted that the transition rate between Fast Recovery/Retransmit and '$l = 0$' states is equal to δ, that is the inverse of a round trip time. Also, when the timeout expires, and the source is ready to resume transmitting, the chain moves from states $s = (0, W_t, 1)$ to $s = (1, W_t, 0)$. In Figure 2 transitions corresponding to timeout expiration are represented by dashed lines. Their rate is equal to τ, where $T = 1/\tau$ is the timeout timer duration. The TCP protocol sets the timeout T on the basis of its own estimation of the average round trip time for the connection. Since the round trip time is a random variable, TCP evaluates it in terms of its estimated average (θ) and standard deviation (let it be denoted with σ). The timeout T is set to $T = \theta + 4\sigma$; however, since the round trip time is modeled as an exponential random variable, the average θ and the standard deviation σ are the same, therefore $T = 5\theta$. In our case we denoted with $1/\delta$ the average round trip time and thus the transition rate τ corresponding to the timeout expiration is $\tau = \frac{\delta}{5}$.

Compound Model Composing the two hierarchical levels described above, corresponding to the application and the transport layers, we can derive a Markovian model whose state space $\mathcal{S}$ is given by:

$$\mathcal{S} = \{I\} \cup \{(w, W_t, l)\}$$

A state $s \in \mathcal{S}$ can be either idle I, or active A. In the latter case, further information is required on the TCP connection, as provided by w, W_t and l.

At connection startup (i.e. when the source becomes active) TCP sets the window size to 1 segment, and the threshold to half the maximum window size; correspondingly, the chain moves with rate α from state I to the state $(1, t_M, 0)$ where t_M is the maximum possible value the threshold may take in our model (see discussion above). For example, in the case of Fig. 2 with $W_M = 16$, we have $t_M = 8$.

When the source switches off, the TCP connection is released; in our model the chain moves with rate β from any active state to the idle state.

Figure 3 sketches the compound model (only some transitions towards state I are shown for the sake of readability).

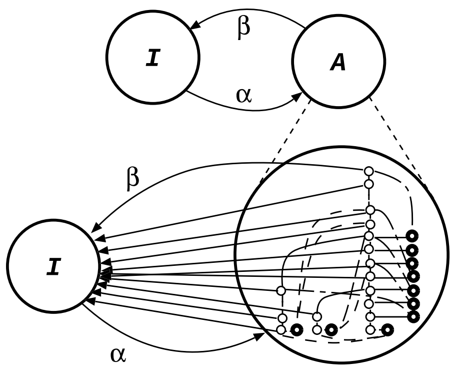


Fig. 3. Model of the source

The Segment Generation Process The segment generation is modeled through a Markov-modulated Poisson process where the Markov chain associated with the source model provides the modulating process.

The value w of the current TCP window size determines the number of segments that can be transmitted during one round trip time. When no losses occur, the time between back-to-back segment generations is an exponential random variable with mean $1/(w\delta)$ so that during one round trip time w segments are transmitted on average.

Eventually, we can compute the average generated traffic λ:

$$\lambda = \sum_{s \in \mathcal{S}} w_s \, \delta \, \pi(s) \tag{2}$$

where w_s is the window size in state s and $\pi(s)$ is the steady-state probability of state s.

3.2 Model of the Network

We assume that the network performance are mainly determined by a single bottleneck. We model the bottleneck as an M/M/1/B queue and use well–known results for such queue to carry out our evaluation. We assume that the input traffic is Poisson with parameter λ which is derived from the analysis of the TCP sources, as explained above. The buffer capacity is equal to B segments. The segment size in bytes S is fixed, so that, given the link capacity C the service time can be deterministically computed as S/C.

Let P be the segment loss probability derived from the solution of the M/M/1/B queue. For every state $s = (w, W_t, l)$ of the Markov chain which describes the TCP behavior, the probability that no losses occur and that one segment is lost are respectively,

$$P_{noloss} = (1-P)^w \quad P_{1loss} = wP\,(1-P)^{w-1}$$

3.3 Summary of Approximations

As stated throughout the description of the model, the complex nature of the TCP protocol has forced us to introduce a number of approximations. We will now review them, briefly addressing their impact on the performance evaluation. First of all, we introduced a few approximations in the source-network interaction:

– the network sees a segment generation process at the sources given by a Poisson distribution; although this might at first appear an excessively optimistic assumption, the aggregation of a large number of On/Off sources can effectively lessen the correlation of the network queue's input process;
– the network model feeds average estimates of loss and delay back to the source model; in turn, the source model uses these average estimates to derive the chain transition rates; therefore, we are assuming that loss rate and delay are exponentially-distributed i.i.d. variables;
– the actual states of sources and network bear no relationship to each other, and only interact through the feedback quantities; again, the inpact of this assumption is softened by the aggregation of many sources.

Furthermore, we simplified the way TCP is commonly implemented:

– we used a limited set of values for the SSthresh (powers of 2 only);
– we assumed fixed-length segments were generated by the sources.
– we assumed that the receiver window never interfered with the sliding window at the transmitter (this is more and more true as the performance of end systems becomes faster and faster).

3.4 Performance Evaluation

The models of TCP connections and of the network explained in previous Sections are independent, stand-alone models. We already explained in Section 2 how the two models cooperate in order to provide a complete description of the system. We will now detail it further.

From the steady-state solution Π of the Markov chain which models the TCP source, we derive the average generated traffic Γ from eq.(2). The aggregate traffic entering the networks is equal to $\lambda = N\Gamma$ where N is the total number of TCP sources, generating traffic Γ.

We analyze the network assuming that the M/M/1/B queue receives an input traffic equal to λ and derive the segment loss probability P and the queueing delay Q to refine our description of the TCP source. P determines the transition rates in the TCP model; Q, instead, is part of the round trip time as in (1). Once we have adjusted these values, we repeat the analysis of the source model to get a new value for Γ, which in its turn is used to refine the network model.

Repeating this procedure until the values of the parameters converge, we can evaluate the stationary behavior of the system. The throughput is given by λ, the goodput by $\lambda \cdot (1 - P)$, while the segment loss probability and the queueing delay are the steady-state values of P and Q respectively.

3.5 Computational Complexity

Since the proposed methodology is iterative, the CPU time (or the number of computations) required for the solution is difficult to predict; furthermore, it depends on the desired accuracy level of the value of the parameters which are iteratively derived. For all the results presented in the following Sections, the number of iteration steps required before the solution could converge to a value with a 10^{-6} accuracy (i.e., relative distance between consecutive estimates), was between 500 and 1000.

Since the solution of the M/M/1/B queue is provided in closed-form, the cost of the solution is dominated by the computation of the steady-state probabilities of the TCP source models. One Markov chain has to be solved for each TCP class. Denoting with M the number of states in the Markov chain which models a class of TCP sources, the solution complexity is $O(\frac{1}{3}M^3)$ using the direct method proposed in [4]. M in its turn depends on the maximum window size and it can be computed, for non-persistent TCP connections, as:

$$M = W_M + \frac{q(q+1)}{2} - 1 + q + \lfloor \frac{W_M}{2} \rfloor - 1 + 1$$

where $q = \left\lceil log_2 \frac{W_M}{2} \right\rceil$ where $\lfloor x \rfloor$ denotes the integer nearest to x. W_M is the maximum congestion window size, $\frac{q(q+1)}{2} - 1$ accounts for the number of states representing the Slow Start mechanism, q accounts for the number of 'timeout' states, $\lfloor \frac{W_M}{2} \rfloor - 1$ accounts for the Fast Recovery states; the last term takes into account the single 'Off' state.

The results presented in the following Section use $W_M = 21$, which entails $M = 38$ accoding to the above equation. The computational cost of the model solution is thus really small, and the actual average time needed to output a generic estimate, say, of TCP throughput in one of the scenarios described above was a few seconds (always less than 10 seconds). This compares to a rough 1,000 seconds necessary for the *ns* simulator to produce the estimate of the same quantity, with the same input values as in the model and with the accuracy reported in Section 4.

4 Results

In this Section, we compare estimates derived from the model with the output of simulations executed using *ns*. All simulations were run until the width of the 98% confidence interval fell within 1% of each point estimate. In all the graphs shown below, solid lines refer to analytical results, dashed lines to simulation results.

We first consider a scenario with 40 ON/OFF sources, the average ON and OFF periods are respectively equal to 100 and 50 ms. The link speed of the bottleneck is 150 Mbps; the segment size is fixed at 1,500 bytes, and the bottleneck queueing capacity is 100 segments; a drop-tail discarding policy is employed. TCP connections are allowed a $W_M = 21$ segments maximum congestion window size.

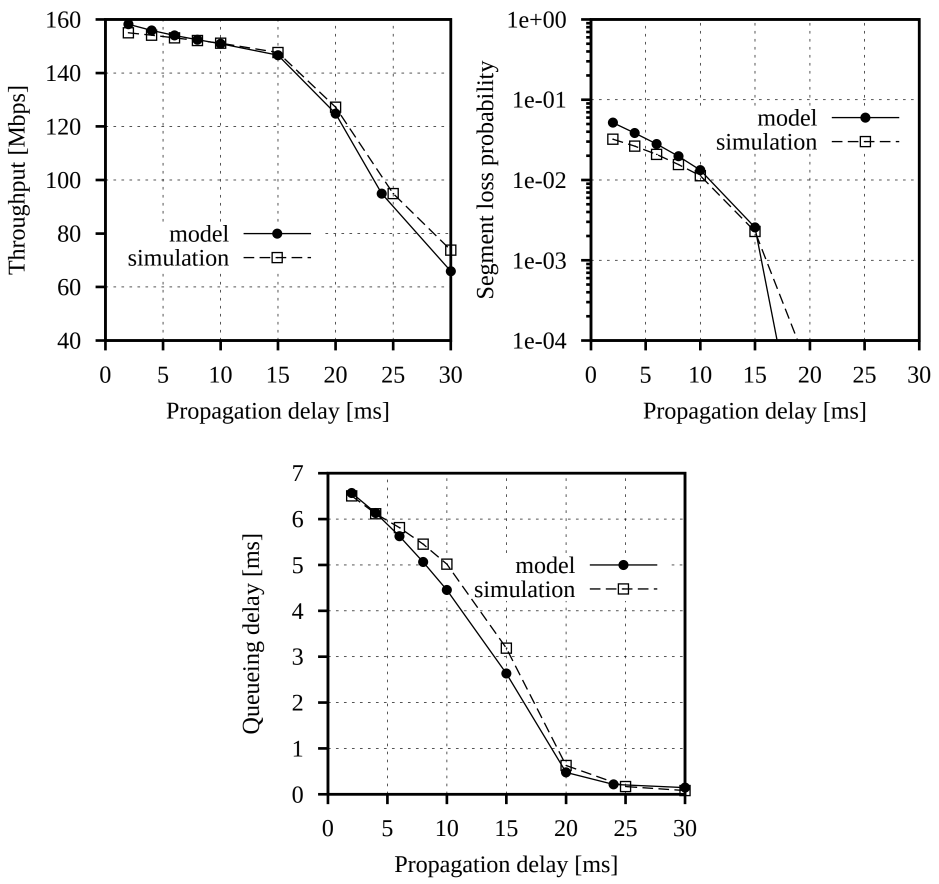


Fig. 4. Throughput, segment loss probability and queueing delay versus two-way propagation delay Q_P; number of connections $N = 40$, $T_{ON} = 100$ ms, $T_{OFF} = 50$ ms

In Fig. 4 throughput, segment loss probability and queueing delay are plotted versus increasing value of the two-way propagation delay. As expected, when the propagation delay increases the throughput decreases; indeed, long distances between transmitter and receiver make the dynamic growth of the window slow down thus reducing the generated traffic. As the generated traffic decreases (for large values of the propagation delay) the segment loss probability and the queueing delay decrease.

Observe that for all the considered performance metrices the analytical predictions are extremely accurate. This is particularly remarkable for segment loss probability which is typically very difficult to predict. In spite of its simplicity, the proposed approach is capable of accurately estimating loss probability even for very low values.

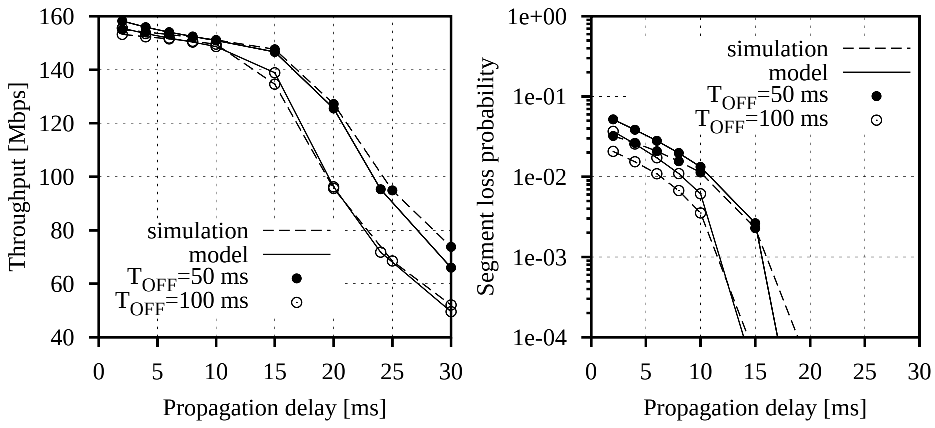


Fig. 5. Throughput and segment loss probability versus two-way propagation delay Q_P; number of connections $N = 40$, $T_{ON} = 100$ ms, $T_{OFF} = 100$, 50 ms

Similar considerations hold for the scenarios presented in Fig. 5 which assess the impact of the source ON-OFF dynamics on the throughput and segment loss probability. We consider two cases with average ON periods equal to 100 ms and different average OFF periods, either equal to 100 or 50 ms. $N = 40$ ON-OFF sources are considered. The case with $T_{OFF} = 50$ ms corresponds to sources which are active most of the time and produce heavy traffic; on the contrary, lighter traffic is due to sources with T_{OFF} =100 ms. The analytical model captures each source type behavior providing very accurate predictions as can be seen from the comparison with simulation results.

In order to investigate the effect of the number of TCP connections, in Fig. 6 we plot the throughtput versus the two-way propagation delay for different values of N: 10, 20, 30, 40. Both ON and OFF periods have mean value equal to 100 ms. Of course, as the number of sources increases, the generated traffic increases. Again, the analytical model provides very accurate predictions.

In Fig. 7 we plot goodput and segment loss probability versus number of sources, for T_{ON} and T_{OFF} respectively equal to 100 and 50 ms. The two-way propagation delay is equal to 4 ms. Both these performance metrics increase with the number of connections, since the generated traffic increases and the network becomes more congested. The accuracy of analytical predictions improves as the number of connections increases; this is justified by the presence of a great number of connections, which makes our statistical independence assumption less and less critical. However, even for a small number of connections, say 20, analytical results are remarkably accurate.

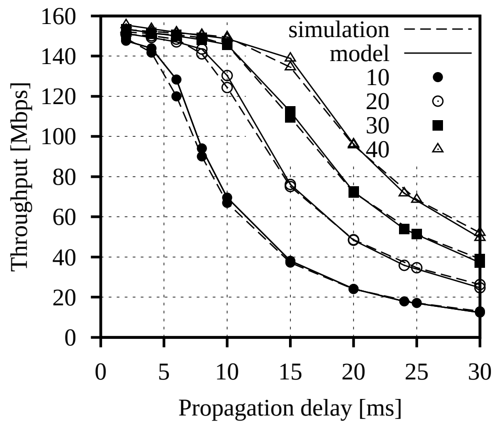


Fig. 6. Throughput versus two-way propagation delay Q_P; number of connections $N = 10, 20, 30, 40$; $T_{ON} = 100$ ms, $T_{OFF} = 100$ ms

5 Conclusion

A methodology to estimate several key performance metrics of TCP Reno was the topic of this paper. Instead of trying to establish a closed-form expression for these metrics, our efforts were directed toward describing an approach that, starting from a Markovian model of TCP source dynamics, investigated the superposition of several such sources; through a procedure that iterates between two complementary models, we predicted the stationary behavior of complex, heterogeneous TCP traffic scenarios.

The results from the analytical approach were validated using simulations, and were found to be very accurate under most circumstances. This may appear surprising given the number of approximation we have introduced. However, we maintain that the aggregation of a large number of On/Off TCP sources lessens the inpact of the correlation introduced by the presence of many flows over a single bottleneck link, making the Poisson assumptions more realistic.

References

1. E. Altman, F. Boccara, J.C. Bolot, F. Nain, P Brown, D. Collange, and C. Fenzy. Analysis of the TCP/IP Flow Control Mechanism in High-Speed Wide-Area Networks. In *34th IEEE Conference on Decision and Control*, pages 368–373, New Orleans, Louisiana, USA, December 1995.
2. C. Casetti and M. Meo. A New Approach to Model the Stationary Behavior of TCP Connections. In *Proceedings of IEEE INFOCOM'2000*, Tel Aviv, Israel, March 2000.
3. K. Fall, , and S. Floyd. Simulation-based Comparisons of Tahoe, Reno, and SACK TCP. *ACM Computer Communications Review*, 26(3):5–21, July 1996.

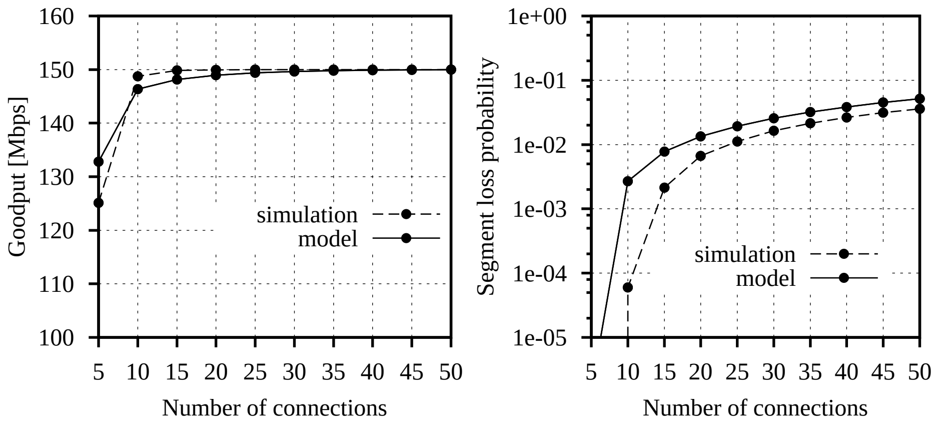


Fig. 7. Goodput and segment loss probability versus number of connections N; Q_P =4 ms; T_{ON} = 100 ms, T_{OFF} = 50 ms

4. W. Grassman, M. Taksar, and D. Heyman. Regenerative Analysis and Steady State Distribution for Markov Chains. *Operation Research*, 33(5):1107–1116, 1985.
5. A. Kumar. Comparative Performance Analysis of Versions of TCP in a Local Network with a Lossy Link. *IEEE/ACM Transactions on Networking*, 6(4):485–498, August 1998.
6. T.V. Lakshman and U. Madhow. The Performance of TCP/IP for Networks with High Bandwidth-Delay Products and Random Loss. *IEEE/ACM Transactions on Networking*, 3(3):336–350, June 1997.
7. M. Mathis, J. Semke, J. Mahdavi, and T. Ott. The Macroscopic Behavior of the TCP Congestion Avoidance Algorithm. *ACM Computer Communication Review*, 27(3):67–82, July 1997.
8. J. Padhye, V. Firoiu, D. Towsley, and J. Kurose. Modeling TCP Throughput: A Simple Model and its Empirical Validation. *Proceedings of the ACM SIGCOMM'98 - ACM Computer Communication Review*, 28(4):303–314, September 1998.
9. V. Paxson. Empirically-Derived Analytic Models of Wide-Area TCP Connections. *IEEE/ACM Transactions on Networking*, 2(4):316–336, August 1994.
10. V. Paxson and S. Floyd. Wide-Area Traffic: The Failure of Poisson Modeling. *ACM Computer Communications Review*, 24(4):257–268, October 1994.
11. W.R. Stevens. *TCP/IP Illustrated, vol. 1.* Addison Wesley, Reading, MA, USA, 1994.

A Markov Model for the Design of Feedback Techniques to Match Traffic Specification Parameters in MPEG Video Sources

Francesco Cocimano, Alfio Lombardo, Giovanni Schembra

Istituto di Informatica e Telecomunicazioni, Università di Catania
V.le A. Doria 6 - 95125 Catania - ITALY
phone: +39 095 7382375 - fax: +39 095 7382397
(fcoci, lombardo, schembra)@iit.unict.it

Abstract. Guaranteeing of quality-of-service (QoS) is a challenging task to promote the evolution of the Internet from a simple data network into a true multiservice network. To this end, the IETF intserv Working Group, with the goal of defining a next generation Internet, has defined two QoS classes: Guaranteed Services and Controlled-Load Services. For both of them the source is required to declare its traffic characteristics by means of a number of *Tspec* parameters and guarantee these traffic characteristics during transmission. The target of this paper is to develop an analytical tool for the design of feedback laws which allow MPEG encoders to inject into the network video traffic shaped according to the declared *Tspec*, while maintaining an acceptable perceived quality.

1. Introduction

In the last few decades progress in the field of digital techniques has led to the development of a variety of multimedia applications. In this context, the MPEG encoding standard [1] has played a key role, becoming the most popular technique in video encoding. The reason for this is the high compression ratio obtained and the flexibility provided by the standard. However, one intrinsic drawback in MPEG encoding is the high burstiness. To overcome this problem a large number of solutions have been proposed in the literature [2-3], all based on some kind of rate control at the output of the encoder. Most encoder control schemes are based on on-line adjusting of the encoder quantizer level through feedback on the quantizer scale parameter [4-5]. Up to now these techniques have been applied to achieve constant bit rate MPEG encoders suitable for video transmission over networks providing users with deterministic bandwidth assignment such as ISDN or satellite networks.

In the next future, the availability of guaranteed services on the Internet [6] will increase the deployment of distributed MPEG video services. In this network scenario, however, the source is required to declare its traffic characteristics by means of a number of *Tspec* parameters and guarantee these traffic characteristics during transmission. In this perspective, choice of the feedback law will constitute a

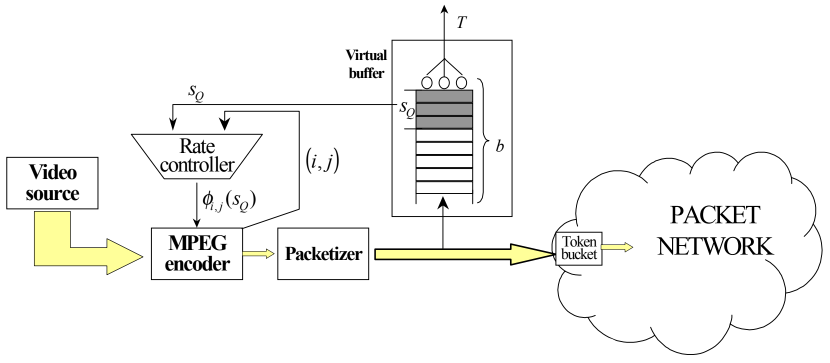


Fig. 1. MPEG video encoder system

challenge to meet the request for rate-controlled MPEG sources coming from the Internet of the next generation. The target of this paper is to define an analytical framework for the design of feedback laws for video traffic shaped according to the *Tspec* declared to the network while maintaining an acceptable image quality.

The analytical framework allows designers to model a rate-controlled MPEG source whatever the feedback law is, provided that the parameter to be varied is the quantizer scale; to this end the framework proposed here takes into account the two parameters used by the most sophisticated rate-controlled sources today [5], that is: a) the state of a counter, incremented by the number of packets emitted by the MPEG encoder, and decreased by a constant number of packets at each time slot; b) the expected activity level of the next frame, in order to estimate the rates produced by the encoding algorithm by means of each potential quantizer scale parameter. In order to compare the efficiency of feedback laws in matching a given traffic specification, the analytical framework evaluates the performance of a rate-controlled MPEG source in terms of both the distortion introduced by the quantization mechanism and the probability of violating the *Tspec* parameters token bucket depth and bucket rate declared to the network [6].

The paper is structured as follows. Section 2 describes the addressed scenario and defines the analytical framework for modeling the rate-controlled MPEG video source; to this end, first the non-controlled MPEG source is modeled as a switched batch Bernoulli process (SBBP) [7-8]. Then, the rate-controlled MPEG source is modeled as a discrete-time queueing system. Section 3 describes how the analytical framework is used to evaluate the performance of a rate-controlled MPEG source. Section 4 considers a feedback law as an example and applies the proposed analytical framework to a case study. Finally, the authors' conclusions are drawn in Section 5.

2. System Description

The system we focus on in this paper is shown in Fig. 1. It is an MPEG video encoder whose output is monitored by a rate controller implementing a feedback law with the target of making the output stream compliant with a number of traffic specification parameters declared to the network, while maintaining an acceptable image quality. In the following we will assume an Integrated Service Internet environment where the traffic specification parameters (*Tspec*) are defined as follows:

- p : peak rate, measured in bytes/sec;
- b : token bucket depth, measured in bytes;
- T : bucket rate of the token bucket, measured in bytes/sec.

These parameters are usually policed by the network at its access point; the parameters b and T are policed by means of a token bucket device. The token bucket is a token pool in which the tokens are generated at a constant rate equal to T bytes/s. The bucket size is of b bytes and represents the maximum capacity of the pool. When an IP packet arrives, a number of tokens, equal to the packet dimension in bytes, is drawn from the pool; if an IP packet arrives when the pool is empty, then the packet is marked as nonconforming to the traffic specification declared by the user. Let $U = 576$ bytes be the packet dimension, comprising 548 bytes of payload at the UDP layer, 8 bytes of UDP header, and 20 bytes of IP header. Let us indicate the peak rate, token bucket depth and bucket rate, expressed in packets/sec, packets and packets/sec, respectively, as $\tilde{p} = p/U$, $K = \lceil b/U \rceil$ and $\tilde{T} = T/U$, where $\lceil x \rceil$ indicates the minimum integer not less than *x*.

In order to match the declared *b* and *T* values, the token bucket device has to be identically reproduced in the source through a virtual buffer, whose state is monitored by the rate controller. The rate controller measures the activity level, i, of the frame which is being encoded and, according to both the frame position in the GoP, j, and the virtual buffer state, s_Q, decides the quantizer scale value to be used to quantize the current frame through a feedback law, $q = \phi_{i,j}(s_Q)$.

3. System Model

In this section we will model the rate-controlled MPEG video encoder system shown in Fig. 1 by means of a discrete-time model, the switched batch Bernoulli process (SBBP) [7-8]. We will use the frame interval, $\Delta \equiv 1/F$, as the time slot. We will indicate the overall emission process as $X(n)$, and we will characterize the GoP structure through the ratio of total frames to intraframes, G_I, and the distance between two successive *P*-frames or between the last *P*-frame in the GoP and the *I*-frame in the next GoP, G_P. For example, in the case of GoP IBBPBB, we have $G_I = 6$ and $G_P = 3$. Moreover, we will decompose $X(n)$ into G_I different emission processes, one for each frame in the GoP, $X_j(h)$, where $j \in J \equiv [1, G_I]$ indicates the frame position in the GoP, and *h* a generic GoP. Of course, we have $X_j(h) = X(h \cdot G_I + j)$. Again, in the case of $G_I = 6$ and $G_P = 3$, $X_j(h)$ will refer to an *I*-frame if $j \in J_I \equiv \{1\}$, a *P*-frame if $j \in J_P \equiv \{4\}$ and a *B*-frame if $j \in J_B \equiv \{2,3,5,6\}$.

The rest of this section is structured as follows. In Section 3.1 we will briefly describe the SBBP process characteristics. Then, in order to obtain the model, as a first step, in Section 3.2, we model the non-controlled MPEG source. Finally, in Section 3.3, the rate-controlled MPEG source will be modeled through a discrete-time queueing system, and the statistics of the output rate will be analytically evaluated.

3.1 Switched Batch Bernoulli Process (SBBP)

An SBBP $Y(n)$ is a discrete-time emission process modulated by an underlying Markov chain. Each state of the Markov chain is characterized by an emission pdf; the SBBP emits data units according to the pdf of the current state of the underlying Markov chain. Therefore an SBBP $Y(n)$ is fully described by the state space $\Im^{(Y)}$ of the underlying Markov chain, the maximum number of data units the SBBP can emit in one slot, $r_{MAX}^{(Y)}$, and the set $\left(Q^{(Y)}, B^{(Y)}\right)$, where $Q^{(Y)}$ is the transition probability matrix of the underlying Markov chain, while $B^{(Y)}$ is the emission probability matrix whose rows contain the emission pdf's for each state of the underlying Markov chain. If we indicate the state of the underlying Markov chain in the generic slot n as $S^{(Y)}(n)$, the generic elements of the matrices $Q^{(Y)}$ and $B^{(Y)}$ are defined as follows:

$$Q_{[s_1,s_2]}^{(Y)} = \lim_{n\to\infty}\left\{S^{(Y)}(n+1) = s_2 \middle| S^{(Y)}(n) = s_1\right\} \quad \forall s_1, s_2 \in \Im^{(Y)} \tag{1}$$

$$B_{[s,r]}^{(Y)} = \lim_{n\to\infty}\left\{Y(n) = r \middle| S^{(Y)}(n) = s\right\} \quad \forall s \in \Im^{(Y)},\ \forall r \in \left[0, r_{MAX}^{(Y)}\right] \tag{2}$$

From $Q^{(Y)}$ we can obtain the steady-state probability array of the underlying Markov chain of the SBBP $Y(n)$, $\underline{\pi}^{(Y)}$, by solving the following linear system:

$$\begin{cases} \underline{\pi}^{(Y)} \cdot Q^{(Y)} = \underline{\pi}^{(Y)} \\ \sum_{s_Y \in \Im^{(Y)}} \pi_{[s_Y]}^{(Y)} = 1 \end{cases} \tag{3}$$

It can easily be demonstrated (see for example [9]) that the autocorrelation function of the above SBBP can be expressed as:

$$R_{YY}(m) \equiv \lim_{n\to\infty} E\left\{Y(n)\cdot Y(n+m)\right\} = \left(\underline{\pi}^{(Y)} \circ \underline{b}^{(Y)}\right)\cdot\left(Q^{(Y)}\right)^m \cdot \left[\underline{b}^{(Y)}\right]^T \tag{4}$$

where the symbol '$\circ$' indicates the element-by-element product, $[\ \cdot\ ]^T$ denotes the transposition operator, and $\underline{b}^{(Y)}$ is the row array containing the mean value of the arrival process in each state, whose generic element, $b_{[s]}^{(Y)}$, is defined as:

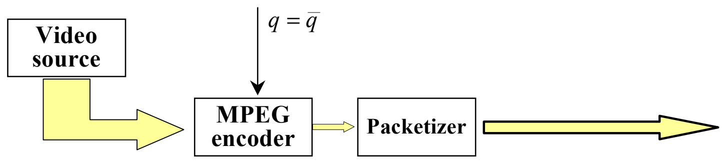


Fig. 2. Non-controlled MPEG encoding system

$$b_{[s]}^{(Y)} = \sum_{r=0}^{r_{MAX}^{(Y)}} r \cdot B_{[s,r]}^{(Y)} \tag{5}$$

Through the spectral decomposition of $Q^{(Y)}$ in (4) we obtain that the autocorrelation function of an SBBP is the sum of a constant term, equal to the square of the mean value of $Y(n)$, and a number W of exponential terms (where W is no greater than the cardinality of $\Im^{(Y)}$), that is:

$$R_{YY}(m) = E^2\{Y\} + \sum_{i=1}^{W} \psi_i \cdot \lambda_i^m \tag{6}$$

where the terms λ_i are the eigenvalues of $Q^{(Y)}$.

Finally, the normalized autocovariance function is simply the sum of the W exponential terms, divided by its maximum value:

$$C_{YY}(m) = \left. \sum_{i=1}^{W} \psi_i \cdot \lambda_i^m \middle/ \sum_{i=1}^{W} \psi_i \right. \tag{7}$$

Below we will introduce an extension of the meaning of the SBBP to model not only an emission process, but also the activity process of an MPEG source. In the latter case we will indicate it as an *activity SBBP*.

3.2 Non-controlled Source Model

In this section we derive the SBBP process $\widetilde{Y}_q(n)$, modeling the emission process of the non-controlled MPEG video source at the packetizer output for a given quantizer scale parameter, q. The non-controlled system modeled in this section is shown in Fig. 2. The model has to capture two different components: the behavior of the *activity process* and the activity/emission relationships. The activity process, $L(n)$, is defined as the average value of the activities in the macroblocks within the frame *n*. The *activity/emission relationships* are defined as the distributions of the sizes of *I*-, *P*- and *B*-frames once the activity *a* of the same frame is given:

$$y^{(I)}(r|a) = \lim_{h\to\infty} \text{Prob}\left\{X_I(h\cdot G_I + j) = r,\ \forall j \in J_I \middle| L(h\cdot G_I + j) = a\right\} \quad \forall a \in A \tag{8}$$

$$y^{(P)}(r|a) = \lim_{h\to\infty} \text{Prob}\left\{X_P(h\cdot G_I + j) = r,\ \forall j \in J_P \middle| L(h\cdot G_I + j) = a\right\} \quad \forall a \in A \tag{9}$$

$$y^{(B)}(r|a) = \lim_{h\to\infty} \text{Prob}\left\{X_B(h\cdot G_I + j) = r,\ \forall j \in J_B \middle| L(h\cdot G_I + j) = a\right\} \quad \forall a \in A \tag{10}$$

where A is the set of possible activity values. After identifying a set $\Im^{(G)}$ of possible activity states, each representing one possible level of scene activity, the transitions between the activity states, and between one frame and the successive one within a GoP, have to be modeled simultaneously. We will obtain the model of the non-controlled MPEG video source in three steps:

1. derivation of an activity SBBP, $G(n)$, modeling the measured activity process, $L(n)$, of the movie. The process $G(n)$ can be represented by its parameter set $(Q^{(G)}, B^{(G)})$. According to [2], the state of the underlying Markov chain of $G(n)$ represents one possible activity state in the set $\Im^{(G)} = \{\text{Very low, Low, High, Very high}\}$;
2. derivation of the SBBP $Y_q(n)$, representing the whole MPEG measured emission process $X(n)$, first calculating its underlying Markov chain from the underlying Markov chain of the activity process $G(n)$, and then the emission process from the activity/emission relationships defined in (8)-(10);
3. derivation of the SBBP $\tilde{Y}_q(n)$ at the packetizer output.

As demonstrated in [8], the desired activity SBBP, $G(n)$, has to fit first- and second-order statistics, $f_L(r)$ and $C_{LL}(m)$, of the activity process $L(n)$. Determination of the activity SBBP $G(n)$ represents a modified version of the so-called *inverse eigenvalue problem*, whose solution can be found in [8].

From the activity SBBP $G(n)$ modeling the activity process, we can derive the SBBP $Y_q(n)$ modeling the non-controlled MPEG emission process when the quantizer scale q is used. To this end let us define the state of the underlying Markov chain of $Y_q(n)$ as a double variable, $S^{(Y)}(n) = (S^{(G)}(n), S^{(F)}(n))$, where $S^{(G)}(n) \in \Im^{(G)}$ is the state of the underlying Markov chain of $G(n)$, and $S^{(F)}(n) \in J$ is the frame position in the GoP at the slot *n*. The resulting set of states of the underlying Markov chain of $Y_q(n)$, $\Im^{(Y)}$, is the Cartesian product of the component subspaces, $\Im^{(G)}$ and J. To calculate the transition probability matrix of the underlying Markov chain of $Y_q(n)$ let us note that the admissible transitions are only between two states such that the frame $S^{(F)}(n+1)$ is the one in the GoP following the frame $S^{(F)}(n)$. Thus the transition probability from the state $S^{(Y)}(n) = (i', j')$ to the state $S^{(Y)}(n+1) = (i'', j'')$, with i', $i'' \in \Im^{(G)}$, and j', $j'' \in J$, is:

$$Q^{(Y_q)}_{[(i',j'),(i'',j'')]} = \begin{cases} Q^{(G)}_{[i',i'']} & \text{if } j''=j'+1 \text{ or } \left(j''=G_I \text{ and } j''=1\right) \\ 0 & \text{otherwise} \end{cases} \quad \textbf{(11)}$$

Let us note that, by what has been said so far, the Markov chain underlying $Y_q(n)$ does not depend on the quantizer scale. For this reason in the following we will indicate its transition probability matrix as $Q^{(Y)}$ instead of $Q^{(Y_q)}$.

As far as the emission probability matrix, $B^{(Y_q)}$, is concerned, its generic element depends on the frame encoding mode, the frame activity, and the quantizer scale used. Once the quantizer scale has been fixed, this matrix can be obtained through the activity/emission relationships. In fact, for a given quantizer scale q when the state of the underlying Markov chain of the activity process is $i \in \Im^{(G)}$, the probability of emitting r packets to encode the frame j is:

$$B^{(Y_q)}_{[(i,j),r]} = \begin{cases} \sum_{\forall a \in A} y^{(I)}(r|a) \cdot B^{(G)}_{[i,a]} & \text{if } j \in J_I \\ \sum_{\forall a \in A} y^{(P)}(r|a) \cdot B^{(G)}_{[i,a]} & \text{if } j \in J_P \\ \sum_{\forall a \in A} y^{(B)}(r|a) \cdot B^{(G)}_{[i,a]} & \text{if } j \in J_B \end{cases} \quad \textbf{(12)}$$

where $y^{(I)}(r|a)$, $y^{(P)}(r|a)$ and $y^{(B)}(r|a)$ are the functions defined in (8)-(10) characterizing the activity/emission relationships, while $B^{(G)}_{[i,a]}$ is the probability that the activity is $a \in A$ when the activity level is $i \in \Im^{(G)}$. The set $\left(Q^{(Y)}, B^{(Y_q)}\right)$ completely characterizes the non-controlled MPEG video source, for a given quantizer scale parameter q.

Finally, the last step is to obtain the MPEG source model at the packetizer output. The objective of the packetizer is to pack bits emitted by the MPEG encoder to obtain packets to be transmitted, according to the UDP/IP protocol suite. Let us denote the packet payload size available to the source to transmit information, expressed in bytes, as $U = 576$ bytes. The arrival process $Y_q(n)$, expressed in bytes/slot, can easily be transformed into an arrival process $\tilde{Y}_q(n)$ in IP packets/slot. In fact, its transition probability matrix and emission probability matrix can be derived as:

$$Q^{(\tilde{Y})}_{[s'_Y, s''_Y]} = Q^{(Y)}_{[s'_Y, s''_Y]} \qquad \forall s'_Y, s''_Y \in \Im^{(Y)} \quad \textbf{(13)}$$

$$B^{(\tilde{Y})}_{[s_Y, \tilde{r}]} = \sum_{r=(\tilde{r}-1)\cdot U+1}^{\tilde{r}\cdot U} B^{(Y)}_{[s_Y, r]} \qquad \forall s_Y \in \Im^{(Y)}, \forall \tilde{r} \in \left[0, r^{(\tilde{Y})}_{MAX}\right] \quad \textbf{(14)}$$

In (14), $r^{(\tilde{Y})}_{MAX}$ is the maximum number of packets needed to transmit one frame and is given by $r^{(\tilde{Y})}_{MAX} = \left\lceil r^{(Y)}_{MAX} / U \right\rceil$ where $\lceil x \rceil$ is the smallest integer not less than x.

3.3 Rate-Controlled Source Model

The rate-controlled encoding system pursues a given target by implementing a feedback law, $q = \phi_{i,j}(s_Q)$, in the rate controller, to calculate the quantizer scale value to be used by the MPEG encoder for each frame. We will indicate the output emission process of the rate-controlled source, expressed in packets/slot, as $\tilde{Y}(n)$, and that of the non-controlled source with a quantizer scale *q* as $\tilde{Y}_q(n)$.

To model the rate-controlled source we have to consider the system shown in Fig. 1 as a whole, indicated here as Σ. We use a discrete-time system model, with a slot duration of $\Delta = 1/F$ equal to the frame duration. The queue of the virtual buffer system is fed by the packets emitted by the MPEG encoder, and is served at a rate equal to the bucket rate, of $\tilde{T}$ packets/slot, declared as a *Tspec* parameter. Moreover, let K be the virtual buffer dimension expressed in packets, another *Tspec* parameter. We assume a *late arrival system with immediate access* [11]: packets arrive in batches, and a batch of packets can enter the service facility if it is free, with the possibility of it being ejected almost instantaneously. Note that in this model a packet service time is counted as the number of slot boundaries from the point of entry to the service facility up to the packet departure point. So, even though we allow the arriving packet to be ejected almost instantaneously, its service time is counted as 1, not 0.

If the serving rate is not expressed in packets/slot, but in bytes/slot, when we convert it into packets/slot we may obtain a $\tilde{T}$ that is not an integer. In this case we will study the queueing system with a service facility serving, in each slot Δ, either $D \equiv \lfloor \tilde{T} \rfloor$ packets with a probability of $p_D = 1-(\tilde{T}-D)$, or $D+1$ packets with a probability of $p_{D+1} = 1-p_D$, where we have indicated the largest integer not greater than x as $\lfloor x \rfloor$. Therefore the number of servers in our queueing system Σ is a random variable with a pdf given by:

$$f_\sigma(d) = \begin{cases} p_D & \text{if } d=D \\ p_{D+1} & \text{if } d = D+1 \\ 0 & \text{otherwise} \end{cases} \tag{15}$$

A complete description of Σ at the n^{th} slot requires a two-dimensional state, $S^{(\Sigma)}(n) = \left(S^{(Q)}(n), S^{(Y)}(n)\right)$, where $S^{(Q)}(n) \in [0, K]$ is the virtual buffer state in the n^{th} slot, i.e. the number of packets in the queue and in the service facility at the observation instant, and $S^{(Y)}(n)$ is the state of the underlying Markov chain of $\tilde{Y}(n)$, but coincides with that of $\tilde{Y}_q(n)$, $\forall q$, given that it is independent of the quantizer scale used. Let us note that, since each observation instant is preceded by departures, the buffer state will never be seen at its maximum value K at the observation instants: more specifically, we have $S^{(Q)}(n) \in [0, K-D]$.

Let us now obtain the rate-controlled encoding system model. To this end we indicate two generic states of the system as $s'_{\Sigma} = \left(s'_Q, s'_Y\right)$ and $s''_{\Sigma} = \left(s''_Q, s''_Y\right)$. We first apply the algorithm introduced in Section 3.2 to obtain an SBBP model of the MPEG video source, $\widetilde{Y}_q(n)$, for each quantizer scale $q \in [1,31]$. So we have a set of parameter sets $\left(Q^{(\widetilde{Y})}, B^{(\widetilde{Y}_q)}\right)$, one for each value of *q*. Then we characterize the emission process of the rate-controlled MPEG video source through a set of matrices $C_{s'_Q}^{(\widetilde{Y}_q)}(r)$, $\forall r \in \left[0, r_{MAX}^{(\widetilde{Y})}\right]$, each representing the transition probability matrix including the probability of *r* packets being emitted when the buffer state is s'_Q. Its generic element can be obtained from the above parameter sets, $\left(Q^{(\widetilde{Y})}, B^{(\widetilde{Y}_q)}\right)$, taking into account that the quantizer scale parameter is chosen by the rate controller according to the feedback function $q = \phi_{i,j}(s_Q)$. We therefore have:

$$\left[C_{s'_Q}^{(\widetilde{Y}_{q''})}(r)\right]_{[(i',j'),(i'',j'')]} = Q_{[(i',j'),(i'',j'')]}^{(\widetilde{Y})} \cdot B_{[(i'',j''),r]}^{(\widetilde{Y}_{q''})} \tag{16}$$

where $q'' = \phi_{(i'',j'')}(s'_Q)$ is the quantizer scale chosen when the source state is $\left(i'', j''\right)$, and the buffer state is s'_Q. Thus, the generic element of the transition matrix of the MPEG encoder system $Q^{(\Sigma)}$ is:

$$Q_{\left[\left(s'_Q,s'_Y\right),\left(s''_Q,s''_Y\right)\right]} = \lim_{n\to\infty} \mathrm{Prob}\left\{S^{(Q)}(n+1) = s''_Q, S^{(Y)}(n+1) = s''_Y \,\middle|\, S^{(Q)}(n) = s'_Q, S^{(Y)}(n) = s'_Y\right\} = \tag{17}$$

$$= \begin{cases} \left(\sum_{r=0}^{s''_Q - s'_Q + D} C_{s'_Q}^{(\widetilde{Y}_{q''})}(r)\right)_{[s'_Y,s''_Y]} + \\ \quad + p_{D+1} \cdot \left(C_{s'_Q}^{(\widetilde{Y}_{q''})}(s''_Q - s'_Q + D + 1)\right)_{[s'_Y,s''_Y]} & \text{if } 0 \le s'_Q \le K - D,\ s''_Q = 0 \\ (p_D \cdot C_{s'_Q}^{(\widetilde{Y}_{q''})}(s''_Q - s'_Q + D))_{[s'_Y,s''_Y]} + \\ \quad + (p_{D+1} \cdot C_{s'_Q}^{(\widetilde{Y}_{q''})}(s''_Q - s'_Q + D + 1))_{[s'_Y,s''_Y]} & \text{if } 0 \le s'_Q \le K - D,\ 1 \le s''_Q \le K - D - 2 \\ (p_D \cdot C_{s'_Q}^{(\widetilde{Y}_{q''})}(K - s'_Q - 1))_{[s'_Y,s''_Y]} + \\ \quad + (p_{D+1} \cdot \overline{C}_{s'_Q}^{(\widetilde{Y}_{q''})}(K - s'_Q))_{[s'_Y,s''_Y]} & \text{if } 0 \le s'_Q \le K - D,\ s''_Q = K - D - 1 \\ \left(p_D \cdot \overline{C}_{s'_Q}^{(\widetilde{Y}_{q''})}(K - s'_Q)\right)_{[s'_Y,s''_Y]} & \text{if } 0 \le s'_Q \le K - D,\ s''_Q = K - D \\ 0 & \text{otherwise} \end{cases}$$

where $C_{s'_Q}^{(\widetilde{Y}_{q''})}(r)$ is a null matrix if $r < 0$ or $r > r_{MAX}^{(Y)}$, and $\overline{C}_{s'_Q}^{(\widetilde{Y}_{q''})}(r)$ is the transition probability matrix including the probability of *at least r packets* being emitted, given by:

$$\overline{C}_{s'_Q}^{(\tilde{Y}_{q''})}(r) = \sum_{p=r}^{r_{MAX}^{(Y)}} C_{s'_Q}^{(\tilde{Y}_{q''})}(p) \tag{18}$$

Once the matrix $Q^{(\Sigma)}$ is known, we can calculate the steady-state probability array of the system Σ as the solution of the following linear system:

$$\begin{cases} \underline{\pi}^{(\Sigma)} \cdot Q^{(\Sigma)} = \underline{\pi}^{(\Sigma)} \\ \underline{\pi}^{(\Sigma)} \cdot \underline{1} = 1 \end{cases} \tag{19}$$

where $\underline{1}$ is a column array all of whose elements are equal to 1, and $\underline{\pi}^{(\Sigma)} = \left[\underline{\pi}_{[0]}^{(\Sigma)}, \underline{\pi}_{[1]}^{(\Sigma)}, \ldots, \underline{\pi}_{[K-D]}^{(\Sigma)}\right]$ is the steady-state probability array, whose generic element, $\underline{\pi}_{[s_Q]}^{(\Sigma)}$, is the array containing the steady-state probabilities of the source aggregate when the virtual buffer is in the state s_Q, that is, the generic term of $\underline{\pi}_{[s_Q]}^{(\Sigma)}$ is:

$$\pi_{[(s_Q, s_Y)]}^{(\Sigma)} = \lim_{n\to\infty} \text{Prob}\left\{S^{(Q)}(n) = s_Q, S^{(Y)}(n) = s_Y\right\} \tag{20}$$

It may be difficult to solve (19) directly since the number of states grows explosively as the counter range increases. Nevertheless, many algorithms [12-13] enable us to calculate the array $\underline{\pi}^{(\Sigma)}$ maintaining a linear dependency on the counter range.

Finally, we can calculate the pdf of the rate-controlled MPEG output traffic $\tilde{Y}(n)$, $f_{\tilde{Y}}(r)$, which is defined as follows:

$$f_{\tilde{Y}}(r) = \lim_{n\to\infty} \text{Prob}\left\{\tilde{Y}(n) = r\right\} \tag{21}$$

From the steady-state solution of the equation system in (19), and applying the theorem of total probability, we have:

$$f_{\tilde{Y}}(r) = \lim_{n\to\infty} \sum_{s_Q=0}^{K-D} \sum_{s_Y \in \Im^{(Y)}} \text{Prob}\left\{\begin{matrix} \tilde{Y}(n) = r, S^{(Q)}(n) = s_Q, \\ S^{(Y)}(n) = s_Y \end{matrix}\right\} = \tag{22}$$

$$= \lim_{n\to\infty} \sum_{s_Q=0}^{K-D} \sum_{s_Y \in \Im^{(Y)}} \text{Prob}\left\{\tilde{Y}(n) = r \left| \begin{matrix} S^{(Q)}(n) = s_Q, \\ S^{(Y)}(n) = s_Y \end{matrix} \right.\right\} \cdot \pi_{[(s_Q, s_Y)]}^{(\Sigma)} = \sum_{s_Q=0}^{K-D} \sum_{s_Y \in \Im^{(Y)}} B_{[s_Y, r]}^{(\tilde{Y}_{\bar{q}})} \cdot \pi_{[(s_Q, s_Y)]}^{(\Sigma)}$$

where $\bar{q} = \phi_{s_Y}(s_Q)$. In addition, from the pdf $f_{\tilde{Y}}(r)$, the average value and variance of $\tilde{Y}(n)$ can also be easily derived as follows:

$$E\{\tilde{Y}\} = \sum_{r=0}^{r_{MAX}^{(\tilde{Y})}} r \cdot f_{\tilde{Y}}(r) \qquad Var\{\tilde{Y}\} = \sum_{r=0}^{r_{MAX}^{(\tilde{Y})}} r^2 \cdot f_{\tilde{Y}}(r) - \left(\sum_{r=0}^{r_{MAX}^{(\tilde{Y})}} r \cdot f_{\tilde{Y}}(r)\right)^2 \tag{23}$$

4. Performance Evaluation

The rate-controlled MPEG encoder system has the following two targets:

1. to keep the output traffic stream $\widetilde{Y}(n)$ compliant with the declared *Tspec*;
2. to maintain an acceptable video quality.

Therefore, the main performance parameters to be taken into account are:

1. the *marking probability*, coinciding with the probability that the token pool in the token bucket saturates; this probability will be derived in Section 4.1 as the loss probability in the virtual buffer of the encoder system, reproducing the token bucket. It is strictly correlated with the choice of the token bucket depth, K, and the bucket rate, $\widetilde{T}$;
2. the *quantization distortion*; this will be derived in Section 4.2 in terms of the pdf's of the PSNR levels.

4.1 Marking Probability

The marking probability, as said previously, can be calculated as the packet loss probability in the virtual buffer when it is loaded by the MPEG encoder output traffic. It can be obtained as the ratio between the average number of IP packets lost in the virtual buffer and the average number of arrived IP packets, that is:

$$P_{\text{Mark}}^{(\widetilde{Y})} = \lim_{n\to\infty} \frac{L^{(\widetilde{Y})}(n)}{N^{(\widetilde{Y})}(n)} = \lim_{n\to\infty} \frac{n}{N^{(\widetilde{Y})}(n)} \cdot \lim_{n\to\infty} \frac{L^{(\widetilde{Y})}(n)}{n} \tag{24}$$

where $L^{(\widetilde{Y})}(n)$ is the cumulative number of packets lost in the discrete-time interval $[0,n]$ and $N^{(\widetilde{Y})}(n)$ is the cumulative number of emissions in the same interval.

The term $\lim_{n\to\infty} N^{(\widetilde{Y})}(n)/n$ represents the per-slot mean emission probability, that is:

$$\lim_{n\to\infty} \frac{N^{(\widetilde{Y})}(n)}{n} = \sum_{s_Q=0}^{K-D} \sum_{s_Y \in \Im^{(Y)}} \sum_{r=0}^{r_{MAX}^{(\widetilde{Y})}} r \cdot B_{[s_Y,r]}^{(\widetilde{Y}_{\bar{q}})} \cdot \pi_{[s_Y]}^{(Y)} \tag{25}$$

where $\underline{\pi}^{(Y)}$ is the array of the steady-state probabilities of the underlying Markov chain of $Y(n)$, obtained in (3), and $\bar{q} = \phi_{s_Y}(s_Q)$.

The term $\lim_{n\to\infty} L^{(\widetilde{Y})}(n)/n$ represents the per-slot loss probability. In order to evaluate this probability, let us note that l packets are lost in the slot n when, indicating the number of packets in the buffer at the observation instant in the same slot as s_Q, $r = (K - s_Q) + l$ packets arrive in the virtual buffer. Therefore:

$$\lim_{n\to\infty} \frac{L^{(\tilde{Y})}(n)}{n} = \sum_{s_Q=0}^{K-D} \sum_{r=K-s_Q+1}^{r_{MAX}^{(\tilde{Y})}} \sum_{l=K-s_Q}^{r} l \cdot \lim_{n\to\infty} \text{Prob}\left\{\widetilde{Y}(n)=r, S^{(Q)}(n)=s_Q\right\} \quad \textbf{(26)}$$

where the term $\lim_{n\to\infty} \text{Prob}\left\{\widetilde{Y}(n)=r, S^{(Q)}(n)=s_Q\right\}$ in (26) can be calculated as follows:

$$\lim_{n\to\infty} \text{Prob}\left\{\widetilde{Y}(n)=r, S^{(Q)}(n)=s_Q\right\} = \sum_{s_Q=0}^{K-D} \sum_{s_Y\in\Im^{(Y)}} \sum_{r=0}^{r_{MAX}^{(\tilde{Y})}} B_{[s_Y,r]}^{(\tilde{Y}_q)} \cdot \pi_{[(s_Q,s_Y)]}^{(\Sigma)} \quad \textbf{(27)}$$

where $\pi_{[(s_Q,s_Y)]}^{(\Sigma)} = \lim_{n\to\infty} \text{Prob}\left\{S^{(Q)}(n)=s_Q, S^{(Y)}(n)=s_Y\right\}$ is calculated as in (19).

4.2 Quantization Distortion

The target of this section is to evaluate the statistics of the quantization distortion, represented here by the PSNR. To this end, we will indicate the distortion curve, representing the distortion obtained when a quantizer scale $q \in [1,31]$ is used to encode the generic frame $\zeta \in \{I,P,B\}$, as $F^{(\zeta)}(q)$. As an example, we have considered one hour of MPEG video sequences of the movie "The Silence of the Lambs". To encode this movie we used a frame rate of $F=24$ frames/sec, and a frame size of $M=180$ macroblocks. The GoP structure IBBPBB was used. The distortion curves obtained for this movie are shown in Fig. 3. In the same figure we have shown. Now let us quantize the distortion curves at L different levels of distortion, $\{\psi_1,\psi_2,\ldots,\psi_L\}$. For example, from a subjective analysis of the considered movie, obtained with 300 test subjects, the following five levels of distortion were envisaged: $\psi_1=[20.6,34.2]$ dB, $\psi_2=]34.2,35.0]$ dB, $\psi_3=]35.0,36.2]$ dB, $\psi_4=]36.2,38.4]$ dB, and $\psi_5=]38.4,52.1]$ dB.

Let $\gamma_l^{(\zeta)}=\{\forall q \text{ such that } F^{(\zeta)}(q)\in\psi_l\}$, for each $l\in[1,L]$, be the range of quantizer scale parameters providing a distortion belonging to the l^{th} level for a frame of the kind $\zeta\in\{I,P,B\}$. Of course, by so doing we are assuming that a variation in q within the generic interval $\gamma_l^{(\zeta)}$ does not cause any appreciable distortion. The quantizer scale interval $\gamma_l^{(\zeta)}$, for each frame encoding mode $\zeta\in\{I,P,B\}$, corresponding to the distortion level ψ_l can easily be achieved thanks to the distortion curves in Fig. 3.

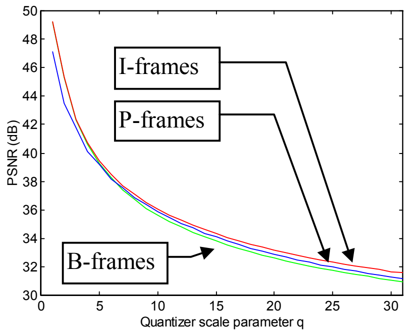


Fig. 3. Distortion curves

Now let us calculate the steady-state probability array of the PSNR levels, whose generic element, $\kappa_l^{(\zeta)}$, represents the probability that the PSNR belongs to the interval ψ_l when the frame ζ is encoded. It can be calculated as follows:

$$\kappa_l^{(\zeta)} = \frac{\underbrace{\sum_{s_Q=0}^{K-D} \sum_{i \in \Im^{(G)}}}_{\phi_{(i,\zeta)}(s_Q) \in \gamma_l^{(\zeta)}} \pi^{(\Sigma)}_{[(s_Q,i,\zeta)]}}{\sum_{s_Q=0}^{K-D} \sum_{i \in \Im^{(G)}} \pi^{(\Sigma)}_{[(s_Q,i,\zeta)]}} \qquad \textbf{(28)}$$

Finally, from (28), the mean PSNR value and its variance can easily be derived.

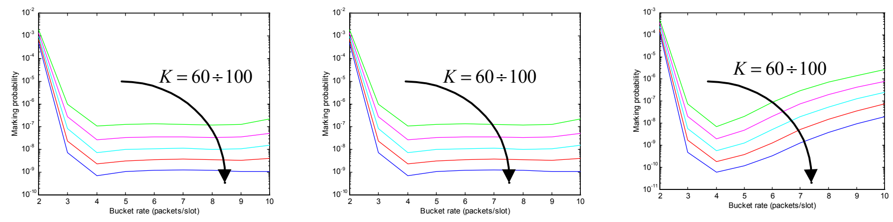
5. Numerical Results

Let us now apply the analytical framework proposed in the previous sections to evaluate the performance of the MPEG encoding mechanism in a real case, encoding of the movie "The Silence of the Lambs".

We assume that, in order to avoid overflow of the virtual buffer representing the token bucket, the MPEG video encoder chooses the quantizer scale value so as to maintain the virtual buffer state no greater than a given threshold τ, chosen in the range of the virtual buffer state, $[0, K]$. More specifically, when the activity state of the video source is *i*, and a frame $\zeta \in \{I, P, B\}$ has to be encoded, the expected number of packets emitted when the quantizer scale value *q* is used is:

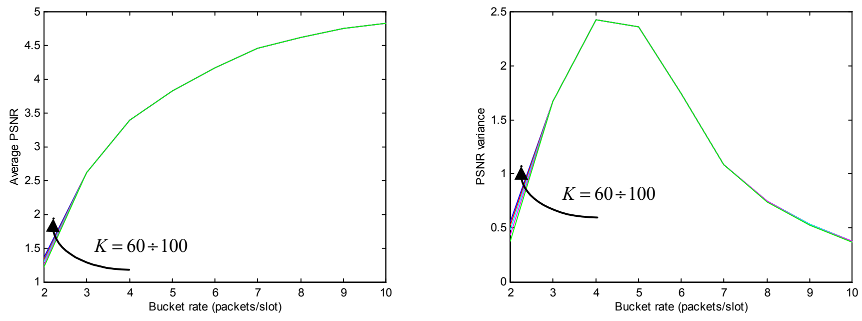
$$R_{i,\varsigma}(q) = \sum_{\forall a \in A} E\left\{y^{(\varsigma)}(r|a)\right\} \cdot B^{(G)}_{[i,a]} \qquad \textbf{(29)}$$

So, if a number of s_Q packets are in the virtual buffer before encoding the j^{th} frame, in order to maintain this number less than or equal to the threshold τ, the rate controller chooses the quantizer scale value according to the following law:



(a): *I*-frames (b): *P*-frames (c): *B*-frames

Fig. 4. Marking probability obtained in the case study



(a): Average value (b): Variance

Fig. 5. PSNR moments obtained in the case study

$$q = \phi_{i,j}(s_Q) \equiv \left[\text{minimum } q \text{ such that: } s_Q + R_{i,\varsigma}(q) \leq \tau \right] \tag{30}$$

Let us carry out the performance analysis by varying the bucket rate $\widetilde{T}$ in the range $[2,10]$ packets/slot, for five different token bucket depths $K \in \{60,70,80,90,100\}$ packets. To this end we have considered a threshold $\tau = 30$.

The marking probability obtained for each kind of frame is shown in Fig. 4, the average value and variance of the PSNR are plotted in Fig. 9, for all the bucket depths considered $K \in \{60,70,80,90,100\}$. In Fig. 9 we can observe that, contrary to expectations, when $\widetilde{T} \geq 4$ packets (high virtual buffer system output rates), all the marking probabilities increase. This is due to the fact that, with higher declared bucket rate values, the quantizer scale value is decreased to increase the average value of the output process; in this way, as desired, the output rate is on average very close to the bucket rate (see Fig. 6a), but, as a consequence, the utilization coefficient of the virtual buffer system approaches unity, and the variance increases rapidly (see Fig. 6b). As is known, these conditions cause a loss probability increase in the virtual buffer.

In Fig. 5 we can observe that, as expected, the image quality increases when the token rate increases, whilst the PSNR variance presents a non-monotonic trend, being

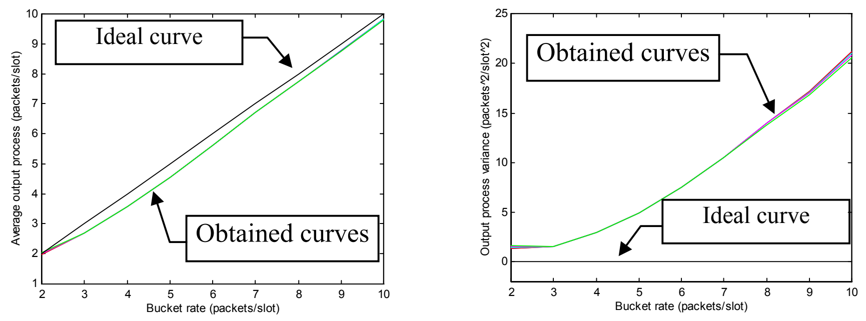


Fig. 6. Output rate moments obtained in the case study

low for both very low and very high bucket rates; this means that for these bucket rate values the image quality is more stable.

Another important consideration can be made by comparing all the above figures: we can deduce that the bucket depth *b* has little influence on the image quality and the output process statistics, while it is decisive for the marking probability.

6. Conclusions

The paper deals with the problem of transmitting MPEG video traffic on the Internet of the next generation, with the constraint of complying with the declared traffic parameters contained in the *Tspec*. To this end, a rate-controlled MPEG encoder has been analytically modeled, in order to provide an analytical tool for the design of feedback laws which allow the MPEG encoder to control the output traffic while maintaining an acceptable perceived quality. More specifically, the analytical framework has been realized first modeling the rate-controlled MPEG video source with a Markov-based discrete-time queueing system, and then analytically deriving its performance in terms of marking probability, output-rate statistics and quantization distortion. Finally, the proposed analytical framework has been applied to a case study.

References

[1] *Coded Representation of Picture and Audio Information*, International Standard ISO-IEC/JTC1/Sc29/WG11, MPEG Test Model 2, July 1992.

[2] A. Puri, R. Aravind, "Motion-compensated video coding with adaptive perceptual quantization," *IEEE Transactions on Circuits and Systems for Video Technology*, vol. 1, no. 4, December 1991.

[3] W. Luo, M. El Zarki, "Quality Control for VBR Video over ATM Networks," *IEEE Journal on Selected Areas in Communications*, vol. 15, no. 6, August 1997.

[4] M. Hamdi, J. W. Roberts, *P*. Rolin, "Rate control for VBR video coders in broad-band networks," *IEEE Journal on Selected Areas in Communications*, vol. 15, no. 6, August 1997.

[5] *Test Model 5*, ISO-IEC/JTC1/SC29/WG11, April 1993.

[6] Paul *P*. White, "RSVP and Integrated Services in the Internet: a Tutorial," *IEEE Communications Magazine*, Vol. 35, No. 5, May 1997.

[7] O. Hashida, Y. Takahashi, and S. Shimogawa, "Switched Batch Bernoulli Process (SBBP) and the discrete-time SBBP/G/1 queue with application to statistical multiplexer," *IEEE Journal on Selected Areas in Communications*, vol. 9, no. 3, April 1991.

[8] A. Lombardo, G. Morabito, G. Schembra, "An Accurate and Treatable Markov Model of MPEG-Video Traffic," *Proc. IEEE Infocom '98*, San Francisco, USA, April 1998.

[9] A. La Corte, A. Lombardo, G. Schembra, "An Analytical Paradigm to Calculate Multiplexer Performance in an ATM Multimedia Environment," *Computer Networks and ISDN Systems*, vol. 29, no. 16, December 1997.

[10] S.-qi Li, S. Park, D. Arifler, "SMAQ: A Measurement-Based Tool for Traffic Modeling and Queuing Analysis Part *I*: Design Methodologies and Software Architecture," *IEEE Communications Magazine*, vol. 36, no. 8, August 1998.

[11] J. J. Bae, T. Suda, R. Simha: "Analysis of Individual packet Loss in a Finite Buffer Queue with Heterogeneous Markov Modulated Arrival Processes: A study of Traffic Burstiness and a Priority packet Discarding," *Proc. IEEE INFOCOM '92*, Florence, Italy, May 1992.

[12] T. Takine, T. Suda, T. Hasegawa, "Cell Loss and Output Process Analyses of a Finite-Buffer Discrete-Time ATM Queueing System with Correlated Arrivals," *Proc. IEEE INFOCOM'93*, San Francisco, CA, March 1993.

[13] A. E. Kamal, "Efficient Solution of Multiple Server Queues with Application to the Modeling of ATM Concentrators," *Proc. IEEE INFOCOM'96*, San Francisco, CA, March 1996.

[14] M. F. Neutz, "Matrix-Geometric Solutions in Stochastic Models: an Algorithmic Approach," Johns Hopkins University Press, Baltimore, MD, USA, 1981.

Intrastandard Hybrid Speech Coding for Adaptive IP Telephony

Francesco Beritelli, Salvatore Casale, Mario Francese, and Giuseppe Ruggeri

Dipartimento di Ingegneria Informatica e delle Telecomunicazioni
University of Catania V.le A. Doria 6, 95125 Catania - Italy
(beritelli, casale, gruggeri)@iit.unict.it

Abstract. This paper presents new speech coding issues and algorithms involved in IP telephony. To develop an efficient system of adaptive voice over IP (VoIP), in fact, besides traditional networking aspects, a series of speech processing issues need to be carefully considered. More specifically, two important aspects of discontinuous transmission are dealt with: the impact of the VAD on source throughput and the need for an efficient system of comfort noise. In addition we propose a variable rate, toll quality CS-ACELP coder that uses coding modes compatible with the three 6.4, 8, and 11.8 kbit/s coding schemes standardised by ITU-T in G.729. In particular, the algorithm presents 4 coding categories, with an average bit rate ranging between about 3 and 8 kbit/s, that adapt the rate to changes in network conditions.

1 Introduction

In the last few years an increasing amount of attention has been paid to technologies for the transmission of voice over data networks. In the next few years a large amount of long-distance telephone traffic will foreseeably be conveyed by IP networks [17]. If voice transmission on IP networks is to be competitive with transmission on PSTN (Public Switched Telephone Networks), users will have to be provided with a quality of service (QoS) that is comparable to that of circuit switching networks. This is not yet possible as there are several problems still to be solved, involving above all the best-effort nature of the IP network protocol. However, to achieve the same level of QoS as PSTN networks, there are, besides networking aspects, a number of speech processing issues that need to be carefully considered. This paper analyses a series of problems linked with variable bit rate (VBR) speech coding in an adaptive IP telephony scenario. A network-driven speech coder, in fact, makes it possible to adapt the rate in relation to the workload on the network, reducing packet loss. Recently a framework for the analysis of adaptive VoIP based on a multistandard speech coding approach was proposed in [8]. In this paper, on the other hand, we propose an intrastandard speech coding solution as it has the advantage of not requiring the integration of a set of coders with different characteristics and of not presenting degradation and latency times during transitions from one coding mode to another. Analysis of the main types of VBR coding shows that the optimal adaptive coding solution in a VoIP scenario is to use a hybrid Multi-Modo/Multi-Rate (M^3R) codec that exploits the robustness to background noise of the multimode technique and the robustness to packet loss of the multirate technique when the network is overloaded. The first aspect is essential given the prospect of integration between traditional telephone networks, both fixed and mobile, and networks based on the IP protocol. Taking in to account this scenario, the paper proposes a toll quality speech coder based on the structure of the CS-ACELP algorithm of G.729 at 8kbit/s [13]. The hybrid coder integrates the two extensions to G.729 at 6.4 kbit/s [9] and 11.8 kbit/s [10] recently standardised by ITU-T, and also exploits new fuzzy pattern recognition techniques for multimode classification [7] and new coding models for the different phonetic classes considered.

2 Network Protocol Issues

Although several ITU-T and IETF study items have recently been devoted to the problem of VoIP, no standard has yet been introduced. Current IP telephony applications therefore refer to the H.323 [15] standard. The main causes of degradation in QoS introduced by the network protocol are: delay, jitter and packet loss. Further to transfer voice over an IP network the RTP-UDP-IP stack of protocols is usually used [22]. These protocols introduce a header of 40 bytes, 12 for RTP, 8 for UDP and 20 for IP. A speech source coded at b byte/s gives rise to a throughput, T, equal to:

$$T = \frac{40}{d_{pacc}} + b \tag{1}$$

where d_{pacc} represents the packeting delay, i.e. the seconds of speech contained in an IP packet. Reducing the rate by a factor r, the throughput will be reduced by a factor R:

$$R = \frac{r}{\frac{40}{d_{pacc}b} + 1} \tag{2}$$

Fig.1 shows the throughput with varying packeting delays and the speech coding standards contemplated by H.323. In the figure we have also considered another two speech coding standards: the 32 kbit/s G. 721 and the 2.4 kbit/s DoD Vocoder. As the packeting delay grows, the throughput tends towards that of the encoder output and the protocol overhead becomes negligible. A codec generally processes a speech signal by splitting it into elementary speech units (frames) lasting L=10 to 30 ms. The packeting delay can therefore be expressed as $N \cdot L + d_l$, where N is the number of frames inserted in a packet, and d_l is the delay related to the look-ahead frame [16]. Unfortunately it is not possible to store an arbitrary number of frames in a single packet, both to avoid an excessive delay and because the loss of a packet would have more serious consequences for the quality of the signal perceived by the user. As Fig. 2 shows, a good trade-off is obtained by using a 40 to 60 ms payload. With lower values, in fact, both the

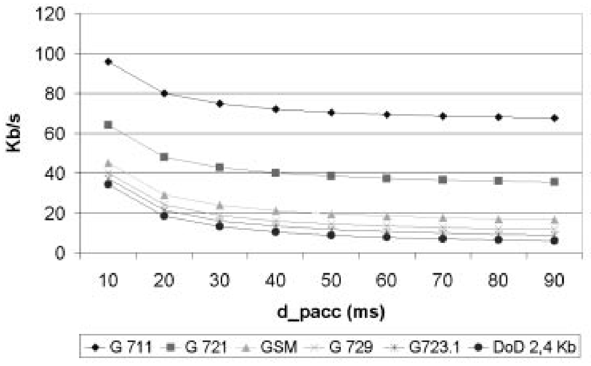


Fig. 1. Throughput versus packeting delay

header and the payload start to have a negative impact on the throughput and with higher values the loss of a packet would considerably reduce the perceived quality. For the sake of simplicity, in Fig. 1 we have plotted the values at 10-ms intervals, even though for some standards, such as GSM, G.723.1 or the 2.4 Kb/s DoD vocoder, packeting values lower than the relative frame lengths, i.e. 20ms, 30ms and 22.5ms, are not considered.

3 Speech Coding Issues

3.1 Silence Suppression

As is well known, one way to reduce the throughput of a speech source, irrespective of the type of coding used, is to implement a discontinuous transmission mode using a Voice Activity Detector (VAD) [17]. More specifically, a VAD identifies periods of silence or background noise during pauses in a conversation. On average for at least 50 % of a conversation no packets are transmitted as the user is listening [14]. Unfortunately the VADs currently available are far from efficient, especially when they are operating in noisy conditions [6]. Fig. 2 shows the throughput measured at the output of a speech source coded using G.729 in the discontinuous mode (DTX) with a rate of 8kbit/s during ON periods and zero kb/s during OFF periods. The three curves refer to the following cases: clean, office noise with an SNR of 20 dB, which is a typical SNR in a quiet office environment, and 10 dB . As can be seen, even a non-critical noise level heavily affects the performance of the VAD. When communications take place in noisy environments, in fact, the network throughput is higher than when there is no noise. Using a VAD that is more robust to background noise, such as the Fuzzy VAD recently proposed in [6], provides greater efficiency in discriminating between active and inactive frames. As shown in Fig. 3, a significant reduction in throughput is achieved, about 8% in average conditions. If we consider (2), we see that to obtain a similar reduction in throughput the source rate would have

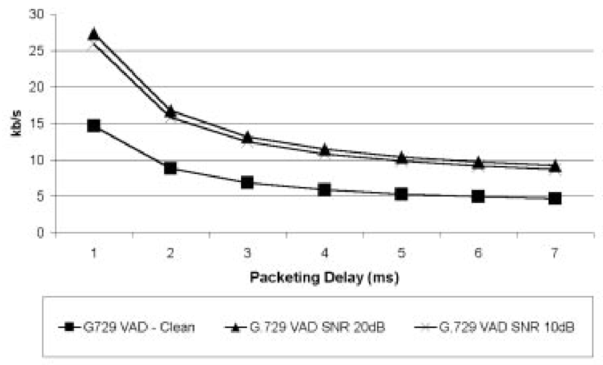


Fig. 2. Throughput at the output of the G.729 annex B codec versus packeting delay in different SNR condition

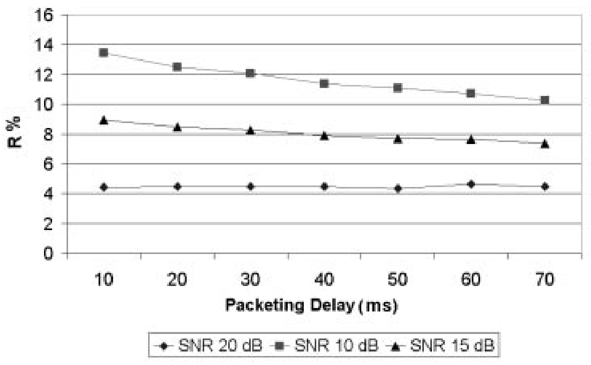


Fig. 3. Throughput reduction factor when Fuzzy VAD is used instead of the standard G.729 VAD versus packeting delay in different SNR conditions

to be reduced by 16% with the substantial difference that, whereas when the VAD is improved the perceived speech quality is relatively unaltered, if the peak rate is lowered the perceived quality drops. Examination of (2) reveals that the lower the peak rate of the initial source, the greater this effect becomes.

3.2 The Issue of Comfort Noise

As no information is transmitted in DTX systems during inactivity periods, a CNG (Comfort Noise Generator) is needed to ensure continuity in the reproduction of background noise. Recently both IETF and ITU-T have realised the need to define a system of Comfort Noise (CN) that will provide an acceptable level of quality even when the codecs used did not originally have a built-in CNG algorithm, e.g. G.711, G.722, G.726, G.727, and G.728. The IETF proposal [21] is to use a CN packet to be sent obligatorily at the beginning of each silence period and optionally at regular intervals established by single applications. In the ITU-T environment, it has repeatedly been remarked that a CN system based on an energy level alone is insufficient. Work has therefore begun on standardising a new CN mechanism that includes in the CN Packet not only information about the level of noise but also other information that allows the noise to be characterised more efficiently, without jeopardising compatibility with the basic version of the packet indicated in [21]. The requirements of the CN system have now been established [1].

3.3 The G.729 Coder

G.729 at 8kbit/s is a recent good quality ITU-T speech coding standard based on conjugate-structure algebraic-code-excited linear prediction (CS-ACELP) [13]. The algorithm is appropriate for multimedia communications and is strongly advised by H.323 for VoIP applications. The algorithm operates on 10ms frames and requires 5ms look-ahead. The input signal is processed with a 140Hz high-pass filter. Then the parameters of the short-term prediction filter, codebook excitation, pitch and the relative gains are determined. These parameters are suitably quantized according to the bit allocation scheme in Table 1. More re-

Parameters for each 10 ms	Annex D	G.729	Annex E Forward	Annex E Backward
LPC	18	18	18	
Pitch period	12	13	13	13
Parity bit	0	1	1+1+1	1+1+1
Codebook	22	34	70	88
Pitch and codebook gains	12	14	14	14
TOTAL	64	80	118	118

Table 1. Bit allocation every 10ms for G.729 and other operative modes.

cently, ITU-T has standardised two extensions of the 6.4 kbit/s and 11.8 kbit/s G.729, respectively indicated as G.729 Annex D [9] and Annex E [10]. The former, which is a low-bit-rate extension of Recommendation G.729, does not offer the same performance as G.729, but in many applications it provides a better quality than the 24 kbit/s Recommendation G.726. As it is an extension, the coding architecture is identical to that of Recommendation G.729. There are, however, certain differences, i.e. the use of a reduced codebook and less fine quantisation of some parameters such as pitch delay and gain (see Table 1). These modifications have reduced the rate from 8kb/s to 6.4 kb/s. The higher-bit-rate extension of G.729 was envisaged for all cases where a better coding quality is required. Here again the basic scheme is the same as that of G.729 but there are some variations that do not only concern the quantisation of the parameters. The most important novelty is the introduction of backward linear prediction analysis, for better coding of music and speech uttered in the presence of stationary noise. This has led to the introduction of two new codebooks, one used in the forward mode and the other in the backward mode. The forward part is identical to Recommendation G.729, while the backward analysis is performed on 30 samples both in coding and decoding. As the LPC (Linear Predicting Coding)coefficients of the backward predictor are not transmitted, the bits saved are used to increase the size of the codebook. One bit is needed to indicate the mode chosen and there is an error control mechanism that uses a parity bit. The differences in bit allocation are summarised in Table 1. The two extensions to G.729 have been a significant reference point for the development of the hybrid multimode/multirate coder presented in the following sections.

4 Adaptive Coding and Control for IP Telephony

4.1 Variable Rate Speech Coding

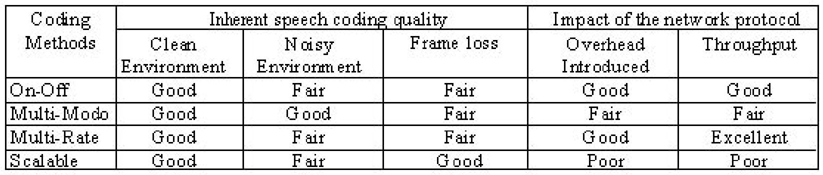
The speech codecs specified by H.323 are all constant bit rate (CBR), i.e. they transmit a given fixed quantity of bits per time unit. A Variable Bit Rate (VBR) speech coder, on the other hand, chooses the most appropriate bit rate from a pre-defined set of operating modes: source- or network-driven [19]. In the former case the codec exploits the various features of speech, choosing the most appropriate coding model for each phonetic class. In the latter case, it is the network itself that chooses the most suitable coding scheme, through a pre-defined rate control mechanism, irrespective of the phonetic contents of the frame. We can distinguish between four different types of VBR speech coding:

– ON-OFF;
– Multimode;
– Multirate;
– Scalable.

ON-OFF transmission takes place with the combined use of a CBR codec and a VAD. In this case the transmission is discontinuous, featuring talkspurt periods (ON) and periods of silence or background noise (OFF) in which the source does not transmit anything. If we further classify the ON and OFF classes into the relative phonetic subclasses (e.g. voiced/unvoiced (ON), stationary/transient noise (OFF), etc.) we obtain a multimode codec that adapts the coding model to the local signal features. According to network conditions, a "multirate" codec chooses one of a certain number of coding schemes with different bit rates but designed to deal with any phonetic class. Finally, a scalable codec uses an embedded structure in which the data packet obtained by coding each single frame comprises a very low bit-rate core (e.g. 2kbit/s) to which a series of enhancement stages are added that increase the bit-rate and therefore the quality of the reconstructed signal. Generally, multirate and scalable coding techniques also adopt a VAD for silence suppression.

4.2 Comparison

In this section we will compare the four techniques outlined above to analyse their behaviour in a VoIP application scenario. More specifically, besides comparing the inherent quality assured by the four different methods, we analysed the impact of the network protocol. Table 2 summarises the comparison, hypothesising



Coding Methods	Inherent speech coding quality			Impact of the network protocol	
	Clean Environment	Noisy Environment	Frame loss	Overhead Introduced	Throughput
On-Off	Good	Fair	Fair	Good	Good
Multi-Modo	Good	Good	Fair	Fair	Fair
Multi-Rate	Good	Fair	Fair	Good	Excellent
Scalable	Good	Fair	Good	Poor	Poor

Table 2. Quality comparison between the various speech coding methods

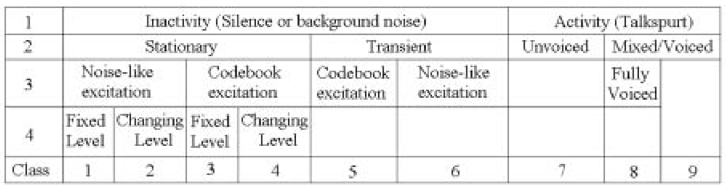
the same average bit rate for all techniques. The assessment scale used (poor, fair, good, excellent) is purely for comparison between the various methods and does not express an assessment of the individual methods. The intrinsic quality of a speech coder is mainly characterised by the quality of the coding algorithm in clean conditions, and its robustness to background noise and frame loss. The impact of the network protocol, on the other hand, is characterised by the overhead introduced by the packeting process and the throughput on the network. In "clean" environments all the methods examined assure a good coding quality, but in the presence of background noise only the multimode technique maintains a good quality. As anticipated in 3.2, in fact, the comfort noise synthesis models currently used by VADs are not capable of guaranteeing satisfactory quality in the reconstruction of background noise and the speech cut by the VAD [3]. Although scalable coding is sub-optimal, it offers the advantage of greater flexibility and robustness to frame loss, as an embedded structure makes it possible to decode the core correctly up to the last stage of enhancement correctly received. With other types of VBR coding, on the other hand, each frame is entirely contained in a packet as so loss of a data packet means that the relative coding frames will be totally lost. In a packetised speech transmission scenario, the advantages of scalable techniques in terms of robustness to frame loss are lost as they require more overhead per speech frame on account of the individual packeting of the various enhancement stages into which a frame is divided. ON-OFF coding systems are less affected by the problem of overhead as they, thanks to the VAD, do not transmit information during silence periods. The multimode technique introduces greater overheads than ON-OFF coding on account of the packeting of non-stationary periods of background noise. Finally, comparing the four methods in terms of throughput,we can observe that the best technique is multirate coding, in that by suppressing silences and adapting the peak bit rate according to network requirements, it reduces the throughput on the network. The multirate technique thus has a better overall impact on the network protocol. From analysis of the Table it also emerges that in a communication scenario in which the IP network is integrated with a wireless network, with users calling from mobile phones, typically in noisy environments, only a hybrid MultiMode/MultiRate technique would be appropriate as would combine the robustness to background noise typical of the multimode technique with the throughput control of the multirate technique. Finally it is reasonable to assume that techniques which introduce less throughput, like the multirate technique, are the most robust to packet loss.

5 Intrastandard Hybrid Coding Proposed

The considerations made in Section 4 show that in a VoIP scenario an optimal solution for adaptive coding can be achieved by using a hybrid Multi-Modo/Multi-Rate (M^3R) codec that exploits the robustness to background noise of the multimode technique and the robustness to packet loss of the multirate technique when the network is overloaded. To guarantee a certain QoS even in critical conditions featuring great delays and background noise, it is necessary to control the peak rate, and therefore use a multirate codec, and also to have good comfort noise models that only multimode coding can provide. The two extensions to G.729, Annex D and Annex E, recently standardised by ITU-T, and a previous work on a robust multimode coder based on the G.729 structure [3], have been the starting point for the development of a hybrid M^3R codec. The new codec guarantees a toll quality, it is robust to background noise, and also has the advantage of being intrastandard, i.e. it fully exploits the architecture of G.729, while maintaining compatibility with both the basic version at 8 kbit/s and the two extesnions at 6.4 kbit/s and 11.8 kbit/s. The codec architecture is based on a modified version of the CS-ACELP algorithm standardised by ITU-T in Rec. G.729. The new features introduced are mainly the insertion of a phonetic classifier that is robust to background noise, on which the multimode part of the codec is based; the addition of the two modes compatible with annex D and E of G.729, with which the coder's bit rate is adapted, and the insertion of new coding models for fully voiced sounds and the synthesis of background noise.

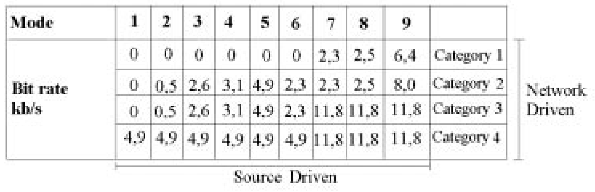
5.1 Robust Phonetic Classification

Phonetic classification is undoubtedly one of the most delicate issues in multimode speech coding [14][19]. The coding of a speech segment by means of an inappropriate model, in fact, causes a degradation in quality, which in some cases reaches unacceptable values. In order to have a speech source driven coder, it needs to exploit fully the large amount of pauses during a conversation and the great variations in the characteristics both of active speech and background noise. The classifier architecture proposed presents four levels of classification characterised by the 9 phonetic classes shown in Figure 4. More specifically, at



1	Inactivity (Silence or background noise)						Activity (Talkspurt)		
2	Stationary				Transient		Unvoiced	Mixed/Voiced	
3	Noise-like excitation		Codebook excitation		Codebook excitation	Noise-like excitation		Fully Voiced	
4	Fixed Level	Changing Level	Fixed Level	Changing Level					
Class	1	2	3	4	5	6	7	8	9

Fig. 4. Phonetic classes considered



Mode	1	2	3	4	5	6	7	8	9	
Bit rate kb/s	0	0	0	0	0	0	2,3	2,5	6,4	Category 1
	0	0,5	2,6	3,1	4,9	2,3	2,3	2,5	8,0	Category 2
	0	0,5	2,6	3,1	4,9	2,3	11,8	11,8	11,8	Category 3
	4,9	4,9	4,9	4,9	4,9	4,9	11,8	11,8	11,8	Category 4

Network Driven
Source Driven

Fig. 5. Coding modes and categories

the first level a Voice Activity Detector (VAD) distinguishes activity segments (talkspurts) from non-activity segments (silence or background noise). At the second level of classification, if the speech segment is active, a voicing detection algorithm discriminates between unvoiced (UV) and voiced (V) sounds. The VAD and V/UV detectors are based on an adaptive version of the algorithms recently proposed in [5] and [4] respectively, where it is demonstrated that they guarantee a greater robustness to background noise than traditional solutions. Both of them use a fuzzy pattern recognition approach exploiting all the information in the input features by means of a set of fuzzy rules automatically extracted by a new hybrid learning tool, based on genetic algorithms and neural networks [20]. In the category of voiced sounds the fully voiced speech segments are identified by a backward cross-correlation algorithm described in [3]. Every frame classified as inactive (silence or background noise) is initially separated into stationary and non-stationary. Here again, the fuzzy pattern recognition approach has proved to be very efficient. In this case we trained a 10-rule fuzzy system with three inputs and one output. The method adopted to identify the optimal set from a total of more than 30 parameters is based on the Fukunaga criterion [12]. All the features selected require a very low computational load and are: the differential power, the differential variance and the differential left variance. As compared with a Quadratic Gaussian Classifier [11] the fuzzy rule-based approach reduces misclassification between stationary and non-stationary noise by more than 10 %. Both stationary and non-stationary frames are closed-loop classified as acoustic noise segments requiring a noise-like or codebook LPC excitation. Finally, within stationary frames, by means of a simple change in level control, the speech segments that maintain their energy level close to that of the previous frame are distinguished from those that have undergone a variation.

5.2 Talkspurts and Background Noise Coding

In order to develop an intrastandard speech coder we used, as the core scheme for the proposed M^3R coder, the structure and the main procedures of the G.729 CS-ACELP algorithm. In particular, we used the standard ITU-T G.729 at 6.4, 8 and 11.8 kbit/s as the coding algorithm for mixed or voiced frames (class 9), whereas for unvoiced sounds we used a simple, time-varying, white Gaussian noise-excited LPC filter (class 7). The noise generator is the same as the one used for the comfort noise system in ITU-T G.729 Annex B. For excitation gain coding we used a non-uniform quantizer with 32 levels, whereas for the other parameters the quantization process was based on the procedures of the G.729 standard. In order to exploit the periodic nature in the steady-state portion of a voiced segment, we used a recent new algorithm for efficient coding of fully voiced sounds at 2.5 kbit/s (class 8) [3]. The method uses a backward cross-correlation measure between the current LPC synthetic residual and the two previous ones. In order to find the most useful way to encode the background noise, we examined a database containing the following 4 examples of stationary noise: Bus, Car, Train, and Dump; and the following 8 examples of non-stationary noise: Office, Restaurant, Street, Factory, Construction, Shopping, Rail station, and Pool. Using a simple coding scheme based on an excited LPC filter, we carried out several subjective tests, coding each type of noise varying the type of excitation and adapting or not adapting the excitation gain and LPC parameters. The

Pulse	Sign	Position
I_0	+1	0,5,10,15,20,25,30,35
I_1	-1	1,6,11,16,21,26,31,36
I_2	+1	2,7,12,17,22,27,32,37
I_3	+1	3,8,13,18,23,28,33,38
	-1	4,9,14,19,24,29,34,39

Table 3. Codebook structure used

codebook structure considered is the same as the one used by the G.729 standard, the only difference being the sign of each pulse, which was fixed a priori according to the position, as shown in table 3. Although this choice assures good quality in the reconstruction of noise, the bit rate is kept below 5 kbit/s. The results of the

Noise	Excitation	LPC	Gain
Bus	Codebook	Stationary	Mixed
Car	Noise-like	Stationary	Mixed
Train	Noise-like	Stationary	Mixed
Dump	Codebook	Stationary	Mixed
Cons	Codebook	Stationary	Mixed
Factory Restaurant	Codebook	Mixed	Non Stationary
Street	Codebook	Mixed	Non Stationary
Office Shop	Codebook	Mixed	Non Stationary
Pool Trains	Codebook	Mixed	Non Stationary

Table 4. Noises characterization

tests are given in Table 4. For a toll quality reconstruction of noise, some types of background noise require a noise-like residual whereas others require codebook excitation. In addition, for some types of noise it is not necessary to update the LPC parameters, and for others again not even the excitation gain. In these cases the bit rate is reduced even further. This acoustical noise characterization highlights the complex, articulated nature of background noise and thus the difficulty of developing efficient comfort noise models. Bearing this in mind, for non-stationary background noise coding we used the unvoiced speech coding model for acoustic noise with noise-like residual (class 6) and a time-varying LPC filter excited by a codebook for acoustic noise characterized by an LPC residual with a non-flat spectrum (class 5). Choice of the type of excitation is made on the basis of a threshold comparison of the ratio between the Weighted Mean Square Errors (WMSE) calculated in the two cases of Gaussian and codebook excitation. For stationary background noise we developed four coding models based on both a similar control performed on the flatness of the LPC residual spectrum, and the changing level of the LPC residual, by means of a simple threshold comparison. Table 5 shows the parameters transmitted and the bit

Mode	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**
LPC	-	-	-	-	18	18	18	18	
Gain	-	5	-	5	5	5	5	-	Look
Codebok	-	-	26	26	26	-	-	-	at
Pitch	-	-	-	-	-	-	-	-	Table
Exitation Lag	-	-	-	-	-	-	-	7	1
Parity	-	-	-	-	-	-	-	-	
Bit/frame	**0**	**5**	**26**	**31**	**49**	**23**	**23**	**25**	

Table 5. Parameters Trasmitted and bit Allocation

allocation for each mode. For the codec to be network-driven as well, we grouped the coding modes in the four categories illustrated in Fig.5 so as to have different average bit rates according to the load on the network. The coding category 2, for average workloads, uses the 8 kbit/s G. 729 as the peak rate. In a situation of network congestion, optimal talkspurt and background noise is forgone in order to minimise the throughput due to the coder. In this case (category 1) no distinction is made between the classes of background noise: they are all reconstructed using a traditional comfort noise system. Active frames are coded as coding category 2 except for mixed sounds, which are coded at 6.4 kb/s as established in G.29 Annex D. Category 3, envisaged for use in low network load conditions, has a higher bit rate as it uses the 11.8 kbit/s extension to G. 729 for activity periods. Finally, category 4, the one with the highest rate, is envisaged for use in very low network load conditions. When the coder operates in this category it selects annex E at 11.8 kb/s for talkspurts, while all frames classified as noise are coded at 4.9 kbit/s using mode 5.

6 Results and Comparison with G.729 *ON/OFF*

During the first series of tests we considered the 4 operating categories and compared them with G.729 at 8 in the discontinuous mode specified in G.729 Annex B [2]. The performance of each mode was evaluated in terms of average bit-rate and perceptive quality. For each mode we carried out a series of tests considering several acoustic noise environments and SNR levels. The speech database used consists of several speech sequences, sampled at 8 kHz, linearly quantized at 16 bits and levelled at 26 dB below codec overload. The sequences, spoken in Italian by male and female speakers, each last 6 minutes with an activity factor of 40 %.

Noise	SNR (dB)	1	2	3	4	5	6	7	8	9
Clean		55.08	3.61	0	0	0	3.22	7.79	8.6	21.7
	00	13.49	9.6	0.26	0.31	0.12	0.79	0.34	3.76	71.19
Car	10	21.72	7.12	0	0	0	0.98	1.83	6.57	61.53
	20	39.64	10.8	0	0	0	1.9	4.8	8.23	34.55
	00	12.23	10.3	13.6	10.51	12.43	4.18	1.09	2.13	33.47
Traffic	10	18.98	13.67	7.11	5.18	9.85	5.53	2.79	4.31	32.55
	20	38.87	11.91	0.15	0.16	0.46	8.43	3.9	7.9	28.09
	00	2.36	1.69	20	15.66	23.04	2.28	1.03	1.15	32.76
Babble	10	2.18	1.52	17.06	13.23	19.73	2.08	2.34	3.9	37.9
	20	15.83	11.07	2.75	2.9	4.26	16.53	4.97	6.03	35.55
Average		22.04	8.13	6.09	4.79	6.98	4.59	3.09	5.26	38.93

Table 6. Percentage of phonetic classes selection in varying acoustic condition

Noise	SNR (dB)	Average rate for M^3R in category 4	Average rate for M^3R in category 3	Average rate for M^3R in category 2	Average rate for M^3R in category 1	Average rate G729/DTX
Clean		7.58	4.58	2.22	1.78	2.52
Car	00	10.08	8.97	5.88	4.65	7.15
	10	9.71	8.30	5.19	4.14	7.36
	20	8.17	5.71	3.18	2.53	6.05
Traffic	00	7.42	5.76	4.19	2.22	5.07
	10	7.63	5.70	3.80	2.25	5.12
	20	7.64	4.99	2.81	2.08	4.69
Babble	00	7.30	6.31	4.86	2.14	4.69
	10	7.94	7.08	5.05	2.57	4.78
	20	8.10	6.29	3.91	2.54	5.18
Average		8.15	6.37	4.11	2.69	5.31
CMOS versus G729/DTX		0.5	0.3	0.1	-0.1	

Table 7. Average bit rate and CMOS

Table 6 shows the percentage of phonetic classes selection while Table 7 shows the average bit-rate, with varying SNRs (0, 10, 20 dB) and types of additive background noise (car, traffic, babble), besides the clean case. The results show how well the phonetic classifier works: for example, the speech coding models developed for background noise with a varying LPC residual spectrum (class 3, 4, and 5) are rarely selected for conversations in the presence of a stationary signal like car noise, whereas with traffic or babble noise they are selected frequently. The opposite happens when classes 1 and 2 are selected. In addition, in noisy environments the percentage of selection of class 6 is considerably reduced, as sounds with a noise-like residual, such as unvoiced sounds, are transformed into sounds requiring a codebook excitation. Naturally, as the average bit rate is closely linked to the behaviour of the phonetic classifier, it depends on both the nature of the background noise and the SNR. The lower bit-rate cases occur in quiet acoustic conditions, as class 1 is chosen more frequently. In the clean case, for example, the average bit-rate is 2.2 kbit/s, for the coding category 2 as class 1 is selected in about 55 % of cases, which corresponds to the percentage of silence frames present in the conversation. The worst case refers to car noise at SNR=0 dB in that, due to the high noise-to-talkspurt misclassifications introduced by the VAD, 71% of the time the codec selects class 9 at the higher rate. A series of informal listening tests based on the Comparison Category Rating (CCR) method [18] were carried out by 24 listeners to evaluate the perceptive quality of the new speech coder. The last row in Table 7 shows the results in terms of Comparison Mean Opinion Score (CMOS) values, i.e. the differences in MOS scores between the coders proposed and the 8 kbit/s G.729 standard with VAD Annex B. We have on average a degradation of 0.1 MOS scores for coding category 1 due to the use of the low bit-rate extension to G.729. Using coding category 2 on average we have an improvement of about 0.1 MOS due to the use of G.729 at 8 kbit/s, the limits of the VAD and the comfort noise system in G.729. Further, with respect to the simple comfort noise system of G.729, the M^3R coder reconstructs the active speech frames detected as noise by means of the most appropriate coding scheme chosen from the six classes developed for background noise. It should be noted that this comparison is a conservative one for the M^3R codec in that there is also a 20 % reduction in bandwidth as compared with the simple ON-OFF coding of G.729 with VAD. Using the high bit-rate extension of G.729 at 11.8 kbit/s (coding category 3) for talkspurt coding the quality improves on average by 0.3 MOS at the expense of a 20% increase in the bit rate. Finally, for coding category 4 there is an improvement of 0.5 MOS as for all types of background noise we use the 4.9 kbit/s codebook excitation mode. In this case there is an increase in bandwidth of about 60%.

7 Conclusions

In this paper we have highlighted some new aspects of speech processing for adaptive IP telephony. These issues are directly connected with the QoS of the communication system and therefore, together with the networking aspects, should be carefully considered in the attempt to bridge the current gap between the QoS of IP telephony and that offered by traditional telephone services. As has been demonstrated, it is important to use a robust voice activity detector not only in relation to the weight it has in reducing voice source throughput but also in relation to the close link with QoS due to the misclassification introduced by current VAD algorithms in the presence of background noise. Together with the VAD, another aspect dealt with is the limits of current comfort noise systems, which are needed to guarantee continuity in the representation of background noise. A characterisation of the main variable rate speech coding techniques showed that the optimal adaptive coding solution is to use a hybrid codec that exploits the robustness to background noise of the multimode technique and the adaptability to variations in network load of the multirate technique. Starting with the CS-ACELP architecture of the G.729 standard at 8 kbit/s and considering the two extensions at 6.4 and 11.8 kbit/s recently standardised by ITU-T as annex D and E to G.729, we have proposed a hybrid intrastandard codec. It can be both source-driven in 9 phonetic classes, and network-driven in 4 coding categories so as to select different average bit rates ranging from about 3 kbit/s to 8 kbit/s.

References

1. Terms of Reference - Comfort Noise for RTP Audio and Video Conferences. Technical report, ITU-T Study Group 16 Q19-21/16, September 1999.
2. Rec. G.729 Annex B. A Silence Compression Scheme for G.729 Optimized for Terminal Conforming. Technical report, ITU-T, Oct. 1996.
3. F. Beritelli. A Modified CS-ACELP Algorithm for Variable Rate Speech Coding Robust in Noisy Environments. *IEEE Signal Processing Letters*, 17(2):31–34, February 1999.

4. F. Beritelli and S. Casale. Robust Voiced/Unvoiced Speech Classification Using Fuzzy Rules. *Proc. Of IEEE Workshop on Speech Coding*, pages 5–6, Pocono Manor, Pensylvania, USA, 7-10 September 1997.
5. F. Beritelli, S. Casale, and A. Cavallaro. A Fuzzy Logic-Based Speech Detection Algorithm for Communications in Noisy Environments. *Proc. of IEEE International Conference on Acoustic Speech and Signal Processing (ICASSP' 98)*, pages 565–568, Seattle, USA, 12-15 May 1998.
6. F. Beritelli, S. Casale, and A. Cavallaro. A Robust Voice Activity Detector for Wireless Communications Using Soft Computing. *IEEE Journal on Selected areas in Communications (JSAC)*, 16(9):1818–1829, December 1998.
7. F. Beritelli, S. Casale, and M. Russo. A Pattern Classification Proposal for Object-Oriented Audio Coding in MPEG-4. *International Journal on Telecommunication Systems*, 9(3-4):375–391, 1998. Special Issue on Multimedia.
8. C. Casetti, J.C. De Martin, and M. Meo. A Framework for The Analysis of Adaptive Voice over IP. To be presented at ICC2000, New Orleans, USA, June 2000.
9. Rec. G.729 Annex D. Annex D to Recomendation G.729: 6.4Kbit/s CS-ACELP Speech Coding. Technical report, ITU-T, Sept. 1998.
10. Rec. G.729 Annex E. Annex E to Recomendation G.729: 11.8Kbit/s CS-ACELP Speech Coding. Technical report, ITU-T, Sept. 1998.
11. Kaled El-Maleh, Ara Samouelian, and Peter Kabal. Frame level noise classification in mobile environments. *Proc. of IEEE International Conference on Acoustic Speech an Signal Processing (ICASSP'99)*, Phoenix, Arizona, May 15-19 1999.
12. K. Fukunaga. *Introduction to Statistical Pattern Recognition*. Academic Press, San. Diego, 1990.
13. Rec. G.729. Coding of Speech at 8 Kbit/s using Conuigate-Structure Algebraic-Codebook-Excited Linear Prediction (CS-ACELP). Technical report, ITU-T, Feb. 1996.
14. A. Gersho. Advances in speech and audio compression. *IEEE Proc.*, 82(6):900–918, June 1994.
15. Rec. H323. Packet Based Multimedia Communication Systems. Technical report, ITU-T, Sept. 1999.
16. M. Hassan and A. Nayandoro. Internet Telephony: Services, Technical Challanges and Products. *IEEE Comm. Magazine*, 38(4):96–103, April 2000.
17. D. Minoli and E. Minoli. *Delivering Voice Over IP Networks*. John Wiley & Sons, 1998.
18. Rec. P.800. Methods for subjective determination of trasmission quality. Technical report, ITU-T, August 1996.
19. E. Paksoy, K. Srinivasan, and A. Gersho. Variable Bit-Rate CELP Coding of Speech with Phonetic Classification. *European Trans. on Telecomunications*, 5:591–601, Sept.-Oct. 1994.
20. M. Russo. FuGeNeSys: A Fuzzy Genetic Neural Systems for Fuzzy Modeling. *IEEE Trans. on Fuzzy Systems*, 6:373–388, Aug. 1998.
21. H. Sculzrinne and S. Caner. RTP Profile for Audio and Video Conferencing with Minimal Control. Technical report, Internet Engineering Task Force, June 1999. This is Work in Progress.
22. H. Sculzrinne, S. Casner, R. Frederick, and V.Jacobson. RTP: A Transport Protocol for Real Time Applications. Technical report, Internet Engineering Task Force, January 1996.

Implementation of a Test-Bed for Telephony over IP: Architectural, Theoretical, and Performance Issues

Marco De Luca[1], Paolo Senesi[1], Francesca Cuomo [2]

[1] CSELT - Centro Studi e Laboratori Telecomunicazioni, Switching Platform Dept.,
Via Reiss Romoli 274, I-10148 Torino, Italy
{marco.deluca, paolo.senesi}@cselt.it

[2] University of Rome "La Sapienza", INFOCOM Dept.
Via Eudossiana 18, I-00184 Roma, Italy
cuomo@infocom.uniroma1.it

Abstract. CSELT is evaluating the feasibility of provisioning Telephony over IP (ToIP) services. ToIP is different from Voice over IP, because it is intended as a Carrier-class service, fully featured and with quality of service guarantees, to be offered to a large number of users, either overlapping or substituting classical Plain Old Telephone Services. CSELT is performing both theoretical and experimental analysis; in particular we are focusing our work on the mechanisms and protocols developed in the standardisation bodies in order to verify how the telephony services can be supported by means of an IP network. In the first half of 2000 we have performed a laboratory trial in order to demonstrate the “proof-of-concept” of ToIP capabilities, performance and interoperability with TDM-based switching technology. In the solution tested in CSELT, provided by a leading manufacturer, the IP technology is deployed in the transit network, so it is necessary to have an interworking function between a traditional circuit network and the IP network. It must be underlined that the solution provided by the manufacturer, as requested by CSELT, was still at an early stage of development, hence it has not been possible to perform a full test campaign. From our theoretical analysis and laboratory experiment we observed the suitability of architectures proposed in the standardisation bodies and by the manufacturers for the provisioning of ToIP services. The experimented system, in particular considering its prototype nature, has shown high performance in terms both of voice and signalling quality. The main problem that arises in ToIP systems is the reliability of the system, that is a fundamental concept for providing carrier-class services. The detailed results of both the theoretical analysis and the laboratory experiment are presented in this paper.

1 Telephony over IP: General Aspects and Motivations

In the last years the voice and data convergence has become more and more important, in particular because of the huge growth of IP traffic and services. In an initial phase all the industry effort has been concentrated on the provisioning of voice services over an IP infrastructure. Voice over IP (VoIP) in this paper is intended as the transport of voice, by means of an IP network without providing a real interworking with traditional PSTN (Public Switched Telephone Network). Examples of application of VoIP are:

- Telcos which use an integrated network to allow corporate customers to interconnect their remote premises for integrated services;
- ISPs, which offer to their customers the possibility to have long-distance call at a cheaper price than in the PSTN.

A quite different and more complex concept, with respect to VoIP, is ToIP (Telephony over IP). ToIP addresses, in addition to voice transport issues, also the control and management plane aspects [1] [2]. ToIP has to be intended as a carrier-class service, that is a real telephony service (fixed and/or mobile), fully featured and with quality of service guarantees comparable to POTS (Plain Old Telephone Service) one's, provided by means of an IP infrastructure. ToIP consists of several components:

- VoIP with quality guarantees;
- Telephony signalling transport with quality guarantees;
- Telephony signalling interworking;
- Supplementary telephone services;
- Integrated Call and Service control;
- Operation, Administration and Maintenance aspects.

Both incumbent operators and new carriers are now facing up to ToIP; the former foresee it as the opportunity to evolve their network infrastructure (i.e. from TDM-Time Division Multiplex to IP) without impacting on the customers while the latter find very attractive the possibility of building from scratch a single network infrastructure for both voice and data traffic.

CSELT is investigating, within a research project named "Packet Telephony", the issues related to the migration from a TDM to an IP network to provide telephony service.

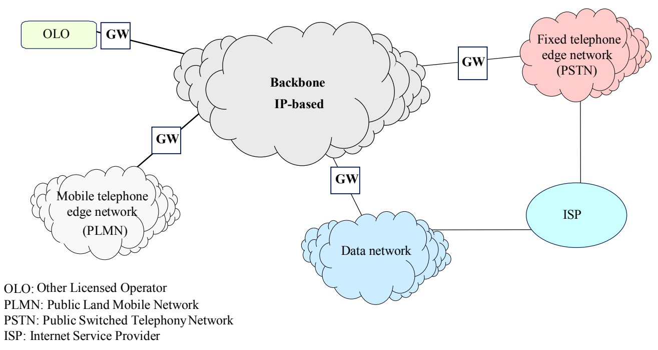

OLO
GW
Backbone
IP-based
GW
Fixed telephone edge network (PSTN)
GW
Mobile telephone edge network (PLMN)
GW
ISP
Data network
OLO: Other Licensed Operator
PLMN: Public Land Mobile Network
PSTN: Public Switched Telephony Network
ISP: Internet Service Provider

Fig. 1. Network scenario

In this context, one of the alternatives considered is the deployment of an IP infrastructure at a transit level; this means a single integrated backbone network carrying voice and data traffic originated by different traditional edge networks, as shown in figure 1. In order to allow communications between the existing edge networks, some interworking functionalities are needed. As a consequence, a new network element, the Gateway (GW), is required to implement all the functionalities to match the requirements of the transport and the control planes for the support of ToIP services.

The aim of the paper is to provide a validation of a ToIP architectural model where both voice and signalling issues are included.

In the first part of this paper (Section 2), an architectural scenario for the ToIP support is presented, focusing the attention on a model of the gateway; in Section 3 a methodology for the evaluation of the components of the Quality of Service (QoS) in a ToIP framework is briefly illustrated. Finally, in Section 4 a test-bed platform, set up at the CSELT laboratories to evaluate the ToIP QoS performance components (as introduced in Section 3), is described in conjunction with the main voice and signalling results derived in the performance analysis.

2 An Architectural Framework

Most of the standardisation bodies and industry fora (IETF [3], ITU-T [4], ETSI [5], MSF [6]) propose a model for the interworking gateway, based on the principle of partitioning the functionalities depending on the network plane (transport, control and management). All the models converge towards the one depicted in figure 2, developed by the IETF SigTran Working Group, which seems to be recognised as a reference model for IP-TDM interworking [3].

This model defines three functional entities:

- *MG (Media Gateway)*: performs packet-circuit switching adaptation. MG terminates SCN (*Switched Circuit Network*) media streams, packetizes the media data and delivers packetized traffic to the packet network; it performs these functions in the reverse order for media stream flowing from the packet network to the SCN;
- *MGC (Media Gateway Controller)*: performs signalling interworking and call control, managing the signalling application protocols; moreover it controls the MG and handles the registration and management of resources at the MG;
- *SG (Signalling Gateway)*: performs transport technology adaptation in the control plane, carrying signalling messages towards MGC. A SG is a signalling agent that receives/sends, without elaborating, SCN native signalling messages at the edge of the IP network. The SG function may also be co-resident with the MG function to process SCN signalling associated with line or trunk terminations controlled by the MG.

The separation between transport (MG) and control (SG+MGC) functionalities leverages the flexibility of the network architecture, that has been one of the drawbacks of traditional telephony switches, because of their monolithic and service embedded structure.

The above separation, in the IETF model, is very rigorous: the MG is emptied from every intelligent functionality, specialising it on lower functionalities as technology adaptation, signal processing and forwarding, thus centralising intelligence in the MGC. In this way it is possible to keep separated media dependent features, strictly related to the technology of the network infrastructure, to media independent ones related to service and control functionalities.

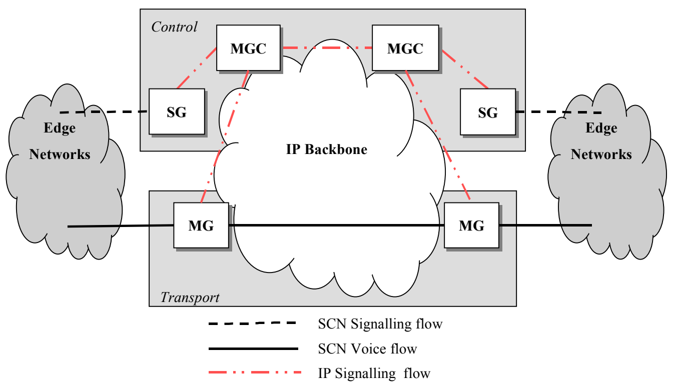


Fig. 2. Reference architectural model

3 Voice and Signalling Requirements for an Assured End-to-End Quality of Service

When evaluating Quality of Service issues, it is very important to separate the aspects related to network performance from the quality perceived by the final users.

The quality of service perceived by end users is "felt" at the end-terminals, which are connected through a network infrastructure whose performance impact the service perception of the end users. Hence it's fundamental to define some objective parameters related to the end users perception and to correlate them to network performance parameters.

The goal of this section is to briefly illustrate the step-by-step methodology of QoS evaluation used in CSELT, that aims at obtaining some reference values, defined for SCN, to be respected by an IP network in order to provide seamless interworking with SCN.

A general approach to the issue of evaluating quality of telephony service, is based on two concepts: *User Perception* and *Network Performance*.

By *User Perception* we mean subjective evaluation by the end users, who estimate the service depending on some indicators; the most significant are:

- *communication voice quality*, related to the user plane;
- *call set-up time*, related to the control plane;
- *serveability*, related to the management plane; it represents the ability of a service to be accessed when requested by the customer and to be provided continuously without particular quality degradation.

All these indicators are conditioned by objectively measurable *Network Performance Parameters*, due to all the operations performed by every single network equipment which could introduce delay, signal corruption and fault.

SCN provides to the end users a high quality and fully-featured telephony service, assuring low transmission delay, consistent availability and performance, and reliable quality of service; all this has been achieved by circuit switching technology deployment and use of a signalling network.

The first step, then, is to provide a mapping between the above-mentioned QoS indicators and SCN network parameters, which can be assigned a reference value to.

As far as the user plane is concerned, the transport of the voice requires constant and low delay but can admit limited frame loss due to the capability of human ears to rebuild corrupted signal; hence, for voice quality the following parameters have been used:

- *End-to-end delay*: amount of delay to transport a voice frame from the originating to the terminating end of the communication. The maximum admitted value, with respect to the interactivity of a communication when echo cancellers are present, is 150 ms as indicated by ETSI [7].
- *Echo*: voice signal attenuation due to signal reflection; in literature, in order to assure acceptable attenuation to the voice signal [8] it is defined the amount of one-way delay permitted. Usually echo cancellers are introduced in the long-distance calls to work out this problem. This parameter has not been measured but evaluated by means of a subjective parameter.

In addition, the intrinsic subjective aspect of voice has pushed for the development of a methodology to evaluate voice quality in a telephony service based on a parameter, named Mean Opinion Score (MOS). It consists in a subjective parameter of the voice quality perceived by users, achieved by computing the mean value of experimental results carried out by a statistically significant group of people, that listen to voice frames and assign a value on a five level degree (5 excellent, 4 good, 3 fair, 2 poor, 1 bad). We used this methodology to evaluate voice quality in the experimental tests, presented later on.

As far as the control plane is concerned, signalling transport, being intrinsically an asynchronous information exchange, requires no loss and limited delay, but it is not influenced by delay variability; the following parameters can be identified:

- *PDD* (*Post Dialling Delay*): the amount of time elapsed between the sending of the last dial by the calling party and the reception of the tone (ringing or busy) by the called party. Actually SCN performance are typically expressed in terms of cross-switch (or cross-office) transfer times, instead of PDD. Hence, we used Hypothetical Signalling Reference Connection (HSRC) [9], ITU-T cross-switch time requirements [10] and a simple ISDN-SS7 call flow to derive by composition the PDD requirements. Usually a reference value is 3-8 secs, depending on end users distance, even if operational values are quite lower. In the test activities we used mainly the SMTT (Signalling Message Transfer Time) value that represents one–way delay of a signalling message.
- *UCR* (*Unsuccessful Call Rate*): it is the percentage of unsuccessfully established call attempts; this parameter describes both signalling network reliability and availability and, usually, should be around 0,2%.

For *serveability*, related to the management plane, the following parameters have been used:

- *End-to-end block probability* (P_{Block}) intended as the probability to not being able to access to the telephony service; its reference value is 2-5 %.
- *Connection loss probability* (P_{Loss}): intended as the probability, during the conversation phase, to lose the connection for network fault events; the reference value is one connection loss out of $4 \cdot 10^{-4}$ events.

Up to now only SCN parameters have been mentioned; these parameters, of course, are conditioned, in a TDM-IP interworking, scenario by a number of parameters in the IP network. In particular, packet loss, delay and jitter are the most important ones.

In the user plane, packet loss percentage influences deeply voice quality. On this issue many case studies are available but everyone differs from each other; as an example, while the ETSI Project Tiphon sets the limit to 3% in order to have a voice quality comparable to the one provided by SCN, recent studies, performed in CSELT, suggest that loss rate of 1-2% take minimum effects on voice, of 4% have sensible degradation effect on voice quality and over 4% make voice quality unsuitable and unacceptable.

Moreover, relationship between packet loss percentage and voice quality is ambiguous, because, though we observe very low packet loss percentage, bursty loss could make voice quality unacceptable; so it is necessary to be ware to the distribution-function that describes loss probability.

In addition, the quality of voice carried on a packet network is affected by voice encoding (which does not affect signalling). In the last years complex encoding algorithms have been defined [11] [12]; although, those allow a lower bandwidth utilisation than classic PCM (Pulse Code Modulation) [13], they introduce a new constant encoding delay (*look ahead*). Moreover, they are heavily sensitive to packet loss effects, because they implement peculiar interpolation mechanisms which work fine only if all packets have been error-free delivered.

Delays in IP network are both fixed (packetization, propagation, frame serialization) and variable, commonly known as jitter. Fixed delays impact directly on end-to-end delay, but they can be predicted in the worst case conditions.

Also variable delays, introduced on voice packets by IP networks, could condition deeply voice quality; in fact one of the most important motivation to the choice of circuit switching technology was its constant delay guarantees. Jitter, or delay variability, strictly depends on network congestion situation, due to router buffer's saturation implying random discarding action. This behaviour in conjunction with voice quality requirements on constant delay force the introduction, at the egress side of the packet network, of dejitter buffers in order to work out variability effects; the correct buffer dimension to avoid jitter problems has to be a wise compromise between higher values, which, in turn, can introduce a high contribution to the fixed part of delay, and lower values, which do not resolve the problem. Anyway an ETSI Tiphon specification [14] reports 75 ms as the maximum value of dejitter delay for an acceptable voice quality.

As far as the signalling transport is concerned, call set-up delay and UCR could be highly influenced by both packet loss and delay. The former could increase delay values because of the retransmission events of the upper protocol level (namely TCP) and, in some cases, brings to unsuccessful calls because of time-outs; the latter impacts directly on call set-up delay and, if extremely high and variable, could deeply influence UCR, because of time-out events. In the control plane, the use of a reliable protocol, based on retransmission, binds packet loss and delay.

As far as the serveability is concerned, in congestion situations, if the IP delay is as high as to cause the time-out of the call attempt processing, the call is refused; this event could increase the end-to-end block probability.

The above discussion is a proof of the complexity of developing a theoretical methodology to put IP network parameters in close relationship to the quality of service in telephony service. The following table summarises the relationship between user perception indicators and network performance, represented by either end-to-end or IP parameters.

Table 1. Summary of the quality evaluation methodology

			User Percpetion indicators						
			Voice Quality			*Call set-up time*		*Serveability*	
Network Perfomance Parameters	*End-to-end*		End-to-end delay	Echo	MOS	PDD	UCR	P_{Block}	P_{Loss}
	IP	Delay	X	X	-	X	X	X	-
		Loss	-	-	X	X	X	-	-
		Jitter	X	-	X	-	-	-	-

Stated that actual IP paradigm is completely inadequate to match the different requirements of the wide range of applications, as voice, signalling, video and data, mechanisms to supply QoS at IP level are still in development. These could provide enhanced features to make ToIP service feasible for a carrier-class service provider. There are two main approaches to provide IP QoS: over-provisioning or traffic engineering; on the latter we'll concentrate our attention.

First, there have been developed some bandwidth management mechanisms, based on algorithms as fair queuing, random early detection and explicit congestion notification, in order to allow reaction to congestion. In the test-bed we utilised a proprietary implementation of Weighted Fair Queuing algorithm; it consists in the assignment of bandwidth to traffic flows, defining different queues with different priority levels, in order to assure that packet loss are uniformly distributed between ingress flows.

Moreover, in order to foresee and assure certain network performance IETF proposed new models and protocols to match QoS requirements.

First, Integrated Service [15] paradigm is based on the principal of resource reservation, in order to assign resources to single applications, and uses RSVP (Resource reSerVation Protocol) to enable an appropriate communication between routers.

Second, Differentiated Services [16] paradigm is based on the principle of priorization, in order to provide different priority level to best effort traffic; it utilises mechanisms to assign resources in the network (mainly fair queuing routing) and presents classification at the ingress functionality as marking, policing and shaping.

4 Experimentation in the Test-Bed

The goal of laboratory experimentation has been the validation of the architectural model for ToIP, presented in Section 2, and the performance study of an early stage developed system platform, provided by a leading manufacturer. The prototypal aspect of the system allowed a full test-bed campaign on voice and signalling quality, but did not allow to check aspects related to supplementary services and management.

The trial configuration is shown in figure 3.

The main functionalities performed by the equipment are the following:

- *Trunk Access Gateway* (TAG): it performs interworking between PSTN and the IP network in the user plane, executing synchronous-to-asynchronous conversion from TDM circuits to IP packets and vice versa; it corresponds, in IETF reference functional model, to the MG for trunk termination;
- *Signalling Gateway and Media Gateway Controller* (SG/MGC): it performs transport interworking in the control plane terminating layers 1-2-3 of the SS7 stack, translating ISDN User Part (ISUP) to handle call in IP network and it controls also the TAG implementing a proprietary protocol;
- *Call Agent* (CA): it executes call handling, terminating MTP3 and ISUP;
- *Management System* (MS): it performs all the management functions.

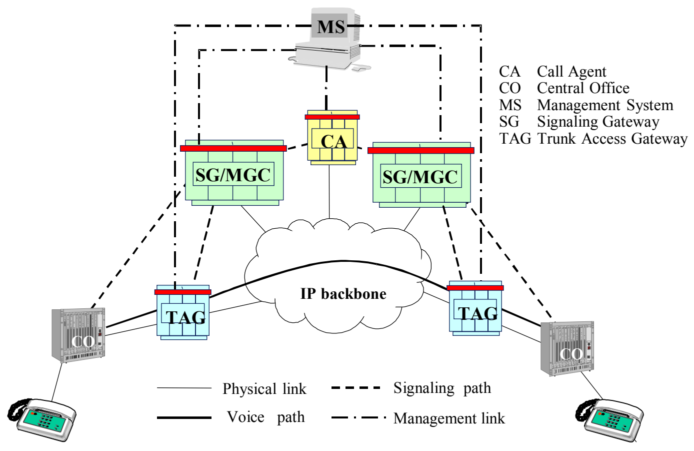


Fig. 3. Test-bed configuration

In order to evaluate completely the performance of the system and end-to-end QoS aspects, different IP backbones have been introduced, identifying three configurations:

A. all equipment were connected by a simple layer 2 switch; one of the main goal of this configuration was to derive the maximum value of signalling load, in terms of BHCA (Busy Hour Call Attempt), supported by the equipment;
B. two IP routers to emulate a simple IP backbone;
C. six IP routers to implement a more complex IP backbone, implementing optionally QoS mechanisms in accordance to the DiffServ paradigm.

In all configurations it has been introduced a bottleneck of 10 Mb/s link for both signalling and voice transport.

In order to perform significant tests for both control and user plane, we injected voice and signalling traffic via PSTN and background data traffic in the IP network.

In the following we define, for signalling and voice quality, measurement parameters, test conditions, configurations and, finally, we present the achieved results.

4.1 Signalling Quality

Signalling messages in the control plane were transported on TCP, which provides a reliable transport service. To evaluate delay impact of TCP under different test conditions, we defined as measurements parameters:

- *Signalling Message Transfer Time* (SMTT), one-way transfer time for signalling messages;
- *Post Dialling Delay* (PDD), as defined in section 3;
- *Unsuccessful Call Rate* (UCR), as defined in section 3.

Initially, in configuration A, we found an optimal signalling traffic value that guarantees the stability of the prototypal equipment: 30 K BHCA (equivalent to 42 messages/second, since in the considered scenario 5 ISUP messages were exchanged to set-up and tear down a call).

The first set of measures have been executed without injecting background traffic in order to evaluate the upper bound values, in all different configurations.

Table 2 resumes mean value, 95 percentile (in accordance to ITU-T recommendations) and variance of the parameters. In addition, statistical distributions of SMTT and PDD are shown, respectively, in figure 4 and 5.

Table 2. SMTT and PDD value in a best-effort IP network without background traffic

		A	B	C
SMTT	Mean [ms]	108	105	112
	95 percentile [ms]	126	145	156
	σ^2 [ms^2]	95	384	530
PDD	Mean [ms]	210	208	205
	95 percentile [ms]	231	247	256
	σ^2 [ms^2]	89	391	604

Table 2 highlights that mean value of SMTT and PDD keeps almost constant on any configuration. This could be explained studying the SMTT's composition which has revealed us that SMTT processing delay is due for 70 % to interworking functionalities, for 30 % to call routing, whereas propagation delay and processing delay due to routers are so small to be not comparable with; so increasing the number of routers does not affect mean values. At the same time, we can observe rising values of 95 percentile and variance in accordance with the number of IP routers; this behaviour is shown also by figures 4 and 5, where it is clearly shown the increase of the spreading in the distribution functions.

Some tests have been performed to investigate how a TCP session, transporting signalling messages, reacts to the injection of another TCP session, transporting background data traffic. This test has been executed, in configuration B, in two different scenarios, as illustrated by figure 6:

1. unidirectional background data traffic TCP injected while signalling traffic is already active;
2. signalling traffic injection while unidirectional background data traffic is already active;

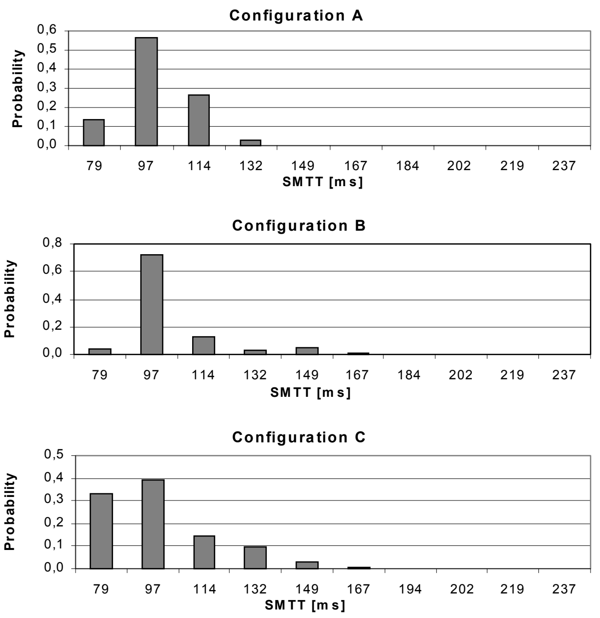


Fig. 4. Statistical distribution of SMTT in a best effort IP network without background traffic

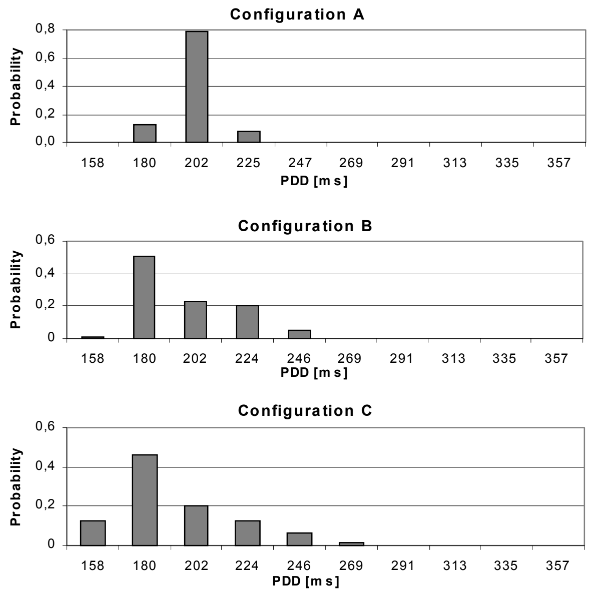


Fig. 5. Statistical distribution of PDD in a best effort IP network without background traffic

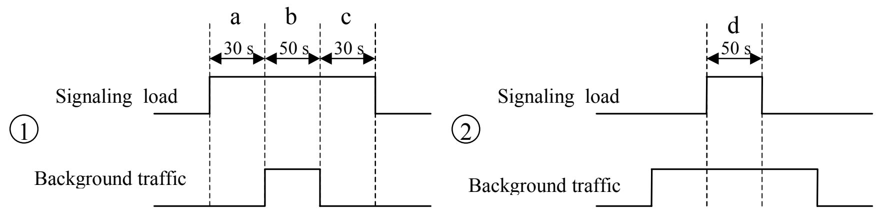


Fig. 6. Traffic curves description

Table 3 reports all parameters values measured, both at the PSTN side and at the IP side. PDD value is not presented as the injected unidirectional background data traffic influenced only SMTT value.

Table 3. Network performance parameters with TCP background traffic injected in an IP network

		a	b	c	d
SMTT	Mean [ms]	106	109	104	115
	95 percentile [ms]	140	149	144	149
IP delay	Mean [ms]	0.31	0.67	0.35	0.67
IP Packet loss	Percentage [%]	0	0	0	2

In the first scenario (1) of this test mean value of SMTT (≅106 ms) did not change during the different phases (a), (b) and (c), giving us the confirmation that different TCP sessions can coexist in IP network thanks to TCP congestion control mechanisms. The 95 percentile value shows how traffic injection cause an acceptable delay increase on signalling messages.

In the second scenario (2), loss of IP packet, transporting signalling messages, has been observed (2%); in this test signalling traffic experienced initial congestion caused by background traffic which, being first injected, took the whole bandwidth; then, both signalling and background traffic began to slow down the transfer rate, due to TCP transmission window reduction mechanism; so the two traffic shared the bandwidth and mean value of SMTT kept unchanged. Anyway, in the second scenario we experienced UCR null, thanks to TCP retransmission mechanism, and a little delay increasing.

Finally, tests have been performed in configuration C, in order to evaluate how UDP background traffic influences signalling performance in a best-effort IP network and how QoS mechanisms could help to match signalling requirements.

As expected, when IP traffic is best-effort, we experienced that, as UDP background traffic has saturated the bandwidth, packet loss occurred and so TCP reacted narrowing its transmission window till performance degradation did not allow signalling messages to be transferred in a reasonable time. This behaviour is clearly shown by the following Table 4, where it is in evidence the amount of mean UDP background traffic injected in the best-effort IP network (it is to be noticed that 10 Mbps is the bottleneck link of the IP network).

Table 4. Network performance parameters with UDP background traffic injected in an IP network without QoS

		0 Mbps	5 Mbps	10 Mbps
SMTT	Mean [ms]	112	119	678
	95 percentile [ms]	156	158	$2 \cdot 10^2$
	σ^2 [ms^2]	530	447	10^6
IP delay	Mean [ms]	1.1	2.5	21
IP Packet loss	Percentage [%]	0	0	5

To work out the problem experienced, we introduced QoS mechanisms in IP network, using a proprietary version of Weighted Fair Queuing (WFQ) and configuring a DiffServ approach, in order to classify signalling traffic with higher priority than background data traffic. In this case, background data traffic injected was 10 Mbps and 12 Mbps in order to force a great amount of loss events. Table 5 shows performance parameters in this test set and highlights how QoS mechanisms allow to match signalling requirements also in highly congestion IP network.

Table 5. Network performance parameters with UDP background traffic in an IP network with QoS

		10 Mbps	12 Mbps
SMTT	Mean [ms]	112	115
	95 percentile [ms]	143	148
	σ^2 [ms^2]	227	254
IP delay	Mean [ms]	3.4	3.4
IP Packet loss	Percentage [%]	0	0

Figure 7 shows SMTT statistical distributions for the case of UDP background traffic of 10 Mbps with and without QoS, in order to highlight the ability of QoS mechanisms to keep low spreading effect on delay.

Figure 8 illustrates a comparison between best-effort and QoS approaches, with the scope to underline how deployment of QoS mechanisms matches the limited signalling delay requirements, also when high UDP traffic (120% of the available bandwidth) is injected in the IP network.

4.2 Voice Quality

Voice quality tests have been performed only in configuration C. During these tests voice frames were transported over RTP/UDP; in order to achieve significant results on data impact to voice quality, we injected both a variable number of simultaneous calls and background data traffic.

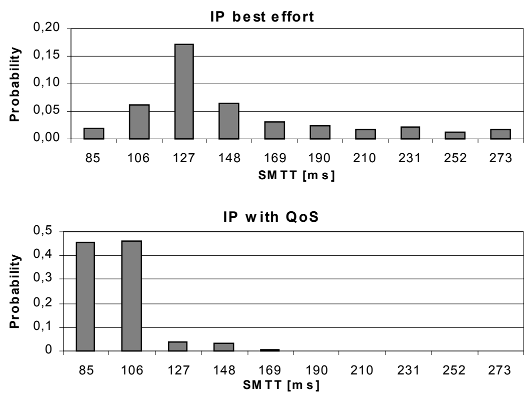


Fig. 7. Statistical distribution of SMTT with UDP background traffic (10 Mbps) with and without QoS

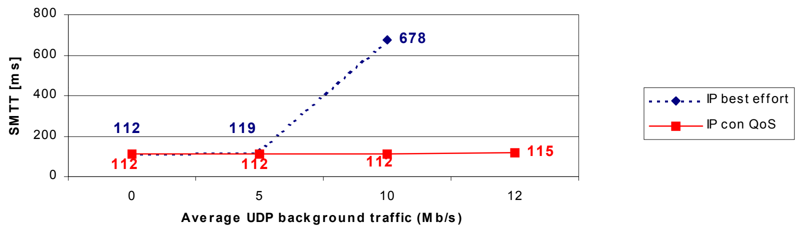


Fig. 8. Signalling performance comparison a best-effort vs. a QoS IP network

The lower bound of the delay that could be achieved is 60 ms, composed by 20 ms due to packetization and 40 ms due to dejitter buffer. Without QoS mechanisms, the performance decreased quickly. In addition, there were higher variance values and significant jitter problems, when background traffic was injected in a best-effort IP network, as depicted in figure 9 (dotted line).

Moreover, delay and packet loss in the IP network are depicted in figure 10 (dotted line). As foreseen in the theoretical discussion in Section 3, it is highly difficult to find a strict correlation between IP and SCN network performance parameters, mainly for the delay one.

Without QoS mechanisms, it was impossible to perform measurements with high volumes of traffic (e.g. 50 calls and 10 Mbps of data) since the high percentage of packet loss (see figures 9 and 10) did not allow to correctly correlate the ingoing packets with the outgoing ones.

Once introduced QoS mechanisms, in the bottleneck link of 10 Mbps was configured 4 Mbps of priority bandwidth for the voice. Figure 9 (full line) highlights that end-to-end delay respects requirements independently by background traffic till voice traffic consumes the provisioned bandwidth, while an immediate degradation was observed as soon as voice traffic exceeds the provisioned bandwidth. This behaviour is also confirmed by observing loss of IP voice-packet, depicted in figure 10 (full line), which shows a sudden rise when we pass 100% of the priority bandwidth allocated.

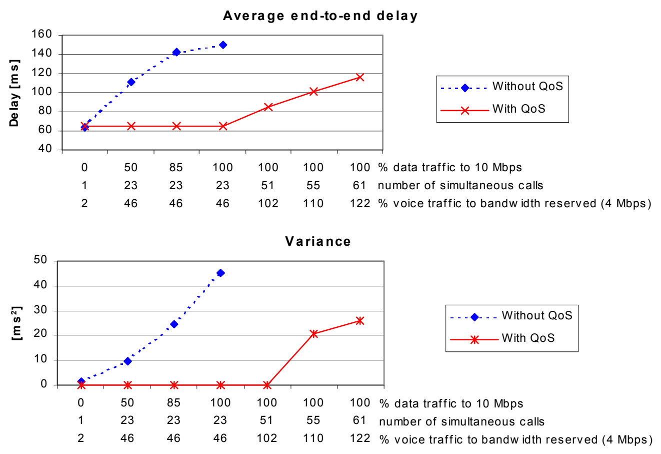


Fig. 9. Average end-to-end delay and Variance

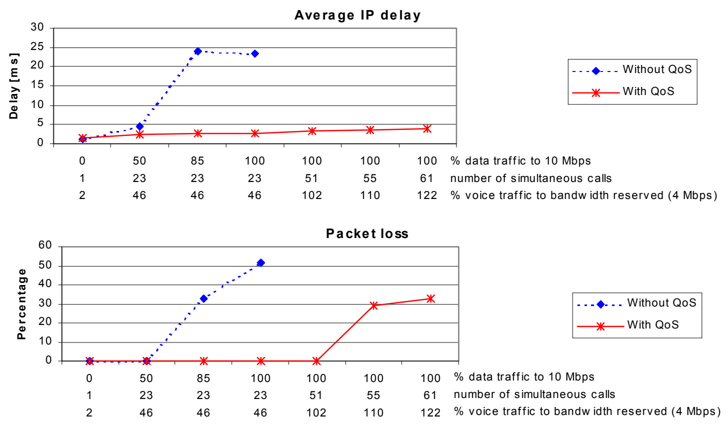


Fig. 10. Average IP delay and Packet loss

Finally, some tests have been performed in order to evaluate MOS parameter presented in Section 3. It is necessary to underline that this subjective measures are not strictly compliant to the methodology standardised by ITU-T, because only integer values have been assigned. Figure 11 confirms that low background traffic injection causes voice quality degradation, when IP is deployed in best-effort, while, when IP is deployed with QoS mechanisms, voice quality keep high independently from the amount of background traffic, but begins to degrade when voice traffic exceeds the maximum provisioned bandwidth.

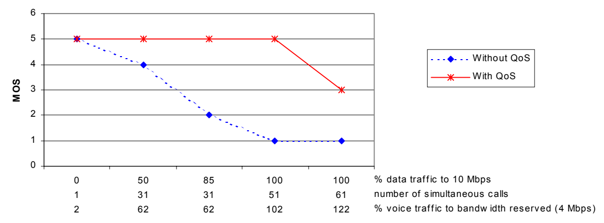


Fig. 11. MOS – Mean Opinion Score

5 Conclusions

From the results achieved in the theoretical analysis and laboratory experiment we proved that the reference models proposed in the standardisation bodies and by the manufacturers are suitable for the provisioning of ToIP services. In fact, from a functional point of view the experimented system, even if in a prototypal stage, satisfied the main requirements of voice and signalling quality. Moreover we defined a rigorous methodology to investigate the performance of ToIP systems and throughout it we proved the necessity of QoS mechanisms to allow data, voice and signalling convergence with respect to telephony high-quality service provisioning. Anyway, our results clearly show that call admission control, provided by the control plane elements, remains a key-feature to avoid the entrance of more voice traffic exceeding the provisioned bandwidth and, consequently, causing the degradation of the telephony service quality, unacceptable from the end-user perception. Finally some issues have arisen regarding the control plane, in particular fault tolerance and security: in this direction we are looking with interest to the work ongoing in the IETF SigTran Working Group, which is studying and defining a new transport layer protocol, named SCTP (Stream Control Transmission Protocol) [17]. This protocol got new features, as message-orientation, stream multiplexing, message bundling, reachability control, multi-homing and security assurance by cryptography methods; from a functional point of view it seems more adequate than TCP for the signalling message transport.

As far as the management and reliability issues are concerned, apart from the experimented system, we have the impression that in order to match PSTN-like requirements, further work is still needed.

Acknowledgements

The authors wish to acknowledge all the CSELT colleagues who have greatly contributed to the work, in particular Maurizio Siviero, Paolo Ferrabone, Gianluca Gritella, Rinaldo Rinaldi, Marco Alfano, Vincenzo Bernardini, Nicola Chiminelli, Raffaele D'Albenzio, Giuseppe De Noia, Enrico Dutto, Roberto Frattini, Paolo Pellegrino, Luca Pesando.

References

[1] M. Hassan, A. Nayandoro, M. Atiquzzaman "Internet Telephony: Services, Technical Challenges and Products" IEEE Communication Magazine, April 2000, pp.96-103

[2] B. Li, M. Hamdi, D. Jiang, Xi-Ren Cao, Y. T. Hou, "QoS-Enabled Voice Support in the Next-Generation Internet: Issues, Existing Approaches and Challenges" IEEE Communication Magazine, April 2000, pp. 54-61

[3] IETF RFC 2719, "Framework Architecture for Signaling Transport", October 1999

[4] ITU-T, Recc. H.323v2 "Packet-based multimedia communications systems", February 1998

[5] ETSI DTS 02003 v0.10.3, "TIPHON Release 3; Network architecture and reference configurations", May 2000

[6] MSF, "MSF System Architecture Implementation Agreement", December 1999

[7] ETSI ETR NA003 - "General aspects of QoS and Network Performance", rev. 1994

[8] ITU-T, Recc. G.131 "Control of talker echo", agosto 1996

[9] ITU-T, Recc. Q.709 "Specification of Signalling System No. 7—Hypothetical Signalling Reference Connection", marzo1993

[10] ITU-T, Recc. Q.725 "Specification of Signalling System No. 7—Siganlling Performance in the Telephone Apllication", marzo 1993

[11] ITU-T, Recc. G.723 "Dual rate speech coder for multimedia communications transmitting at 5.3 and 6.3 kbit/*s*", marzo 1996

[12] ITU-T, Recc. G.729 "Coding of speech at 8 kbit/s using conjugate structure algebraic code excited linear prediction (CS-ACELP)", marzo 1996

[13] ITU-T, Recc. G.711 "Pulse Code Modulation (PCM) of voice frequencies"

[14] ETSI DTR/Tiphon-05001 v1.2.5, "General aspects of QoS", settembre 1998

[15] IETF RFC 2205, "Resource ReSerVation Protocol (RSVP) - Version 1 Functional Specification", settembre 1997.

[16] IETF RFC 2475, "An Architecture for Differentiated Services", dicembre 1998.

[17] R. R. Stewart et al., "Simple Control Transmission Protocol - draft-ietf-sigtran-sctp-13", July 2000

Enhanced Weighted Round Robin Schedulers for Bandwidth Guarantees in Packet Networks

Andrea Francini[1], Fabio M. Chiussi[1], Robert T. Clancy[2], Kevin D. Drucker[1], and Nasser E. Idirene[1]

[1] Bell Laboratories, Lucent Technologies
Holmdel, NJ 07733, U.S.A.
{francini,fabio,kdrucker,nei}@lucent.com
[2] Currently with Sycamore Networks, Wallingford, CT 06942, U.S.A.
robert.clancy@sycamorenet.com

Abstract. Because of their minimal complexity, Weighted Round Robin (WRR) schedulers have become a popular solution for providing bandwidth guarantees to IP flows in emerging networks that support differentiated services. The introduction of applications that require flexible bandwidth management puts emphasis on hierarchical scheduling structures, where bandwidth can be allocated not only to individual flows, but also to aggregations of those flows. With existing WRR schedulers, the superimposition of a hierarchical structure compromises the simplicity of the basic scheduler. Another undesirable characteristic of existing WRR schedulers is their burstiness in distributing service to the flows. In this paper, we present two enhancements for WRR schedulers which solve these problems. In the first enhancement, we superimpose a hierarchical structure by simply redefining the way the WRR scheduler computes the timestamps of the flows. This "soft" hierarchy has negligible complexity, since it does not require any additional scheduling layer, yet is highly effective. The second enhancement defines an implementation of a WRR scheduler that substantially reduces the service burstiness with marginal additional complexity.

1 Introduction

A crucial issue in emerging networks supporting differentiated services is the provision of bandwidth guarantees to IP flows. As more elaborate applications requiring flexible bandwidth management become popular, the focus is on the ability of segregating bandwidth to individual flows as well as to aggregations of those flows in a hierarchical manner. The challenge is to design schedulers that support such functionality with very low implementation cost.

Some of the existing packet schedulers offer good worst-case delay performance in addition to providing accurate bandwidth guarantees [1, 4, 5]. However, in networks where packets have variable size, as is the case in IP, the implementation cost of schedulers with good worst-case delay properties is substantial. For example, the scheduler presented in [5], which sorts the flow timestamps using a two-dimensional calendar queue, still requires several thousands of sorting bins to indeed achieve tight delay bounds.

Because of the heavy implementation cost of packet schedulers that achieve optimal delay performance [4, 5], and because worst-case delay performance is actually rather secondary to robust bandwidth performance in IP networks, we restrict the scope of this paper to schedulers that have very low complexity and can provide robust bandwidth guarantees, but do not necessarily achieve delay bounds that are close to optimal. In particular, we focus on the family of *Weighted Round Robin* (WRR) schedulers [6, 7, 8]. Different instantiations of these scheduling algorithms have appeared in literature; well known examples are the *Deficit Round Robin* (DRR) [7] and the *Surplus Round Robin* (SRR) [8] algorithms. Their delay properties are far from optimal, but they enforce strict bandwidth guarantees with minimal implementation cost.

Existing WRR schedulers suffer from two drawbacks. First, they are essentially "single-layer" schedulers, in that they can provide bandwidth only to individual flows. Superimposing multiple scheduling layers, and thus implementing a hierarchical and flexible structure that can not only allocate bandwidth to individual flows, but also create aggregations of flows and segregate bandwidth accordingly, compromises the simplicity of these schedulers. Second, WRR schedulers distribute service to the individual flows in a highly-bursty manner, since they continue servicing a flow until its allocated share within a service frame is exhausted. Reducing such service burstiness is highly desirable, since it leads to lower losses, allows for more effective buffer management, and improves the delay distribution. In this paper, we propose two enhanced WRR schedulers to solve these problems. Both enhancements use the SRR algorithm, which proves more suitable than DRR to segregate bandwidth with minimal overhead.

In our first enhancement, we create a WRR scheduler with a hierarchical structure for bandwidth segregation. Our objective is to provide bandwidth guarantees to aggregations of flows (which we call *bundles*) as well as to individual flows in a completely transparent manner, *i.e.,* without using any additional scheduling structure. We achieve our objective by first defining a timestamp-based version of the SRR algorithm, and then simply enhancing the way the timestamps are manipulated. The resulting scheduling hierarchy is "soft", and therefore has negligible complexity; yet, it is effective, as we demonstrate with both theoretical arguments and experimental results. Our hierarchical structure provides bandwidth segregation to the bundles as well as robust guarantees to the individual flows.

Our second enhancement consists of an implementation of the SRR scheduler which is based on a modified calendar queue that reduces the service burstiness. A substantial improvement in burstiness is achieved with a very small number of sorting bins in the calendar queue; thus, the overall complexity of the scheduler increases only marginally.

The rest of this paper is organized as follows. In Section 2, we briefly recall some general scheduling concepts, review the calendar queue, which we use as the starting point for our implementation of the WRR with reduced burstiness, and describe the DRR and SRR algorithms. In Section 3, we present our enhanced WRR scheduler to provide soft bandwidth segregation, and show simulation results that confirm the operation of the scheduler. In Section 4, we present an implementation of our enhanced WRR scheduler to reduce the service burstiness, together with supporting experimental results. Finally, in Section 5, we offer some concluding remarks.

2 Background

2.1 Packet Schedulers with Allocation of Service Shares

The system that we address in this paper is a packet multiplexer where multiple traffic flows contend for the bandwidth that is available on the outgoing link. The multiplexer maintains a distinct queue of packets for each flow (*per-flow queueing*), and handles the flows as individual entities in the scheduler that distributes access to the link (*per-flow scheduling*).

We focus on the scheduling part of the system, and restrict our attention to schedulers that associate a service share ρ_i to each configured flow i and always distribute service to the flows in proportion to their allocated shares (popular shared-based service disciplines include *GPS-related* [1, 9, 10, 11] and WRR [6, 7] schedulers). The *latency* θ_i experienced by flow i captures the worst-case delay properties of a share-based scheduler, provided that the sum of the allocated service shares does not exceed the capacity of the server. [1]

Packet-by-packet Rate-Proportional Servers (P-RPS) [1] feature minimum latency among GPS-related schedulers. GPS-related schedulers that are not P-RPS, such as *Self-Clocked Fair Queueing* (SCFQ) [11] and *Start-time Fair Queueing* (SFQ) [12], as well as WRR schedulers [6, 7], have much looser latency guarantees than any P-RPS [2]. However, their implementation complexity is much lower, and their bandwidth guarantees are still robust, which makes them extremely appealing whenever the flows to be scheduled have no stringent delay requirements.

2.2 The Calendar Queue

Most schedulers with allocation of service shares assign *timestamps* to the backlogged flows to determine the order of transmission of the respective packets. A timestamp generally expresses the service deadline for the associated flow, in a *virtual-time* domain that is specific of each algorithm.

The implementation cost of a timestamp-based scheduler is commonly dominated by the complexity of sorting the timestamps. The *calendar queue* [5, 13, 14, 15, 16] reduces the complexity of the sorting task by providing an ordered structure of bins, each bin being associated with a certain range of timestamp

[1] The latency of flow i is defined as the maximum interval of time that the flow must wait at the beginning of a busy period before its service rate becomes permanently not lower than the allocated service share ρ_i [2].

values. The *representative timestamp* of a bin is given by the lower end of the timestamp range associated with the bin. The bins are ordered in memory by increasing value of their representative timestamps. Flows (or packets) are stored in the bins based on their current timestamps. The bins are then visited in their order in memory. By construction, when the mechanism visits a bin that is non-empty, the representative timestamp of the bin is the minimum representative timestamp in the calendar queue. The whole idea is to provide a structure where each position in memory has a direct relation with the value of the timestamps, and simplify the sorting task by exploiting the spatial separation that is achieved by associating the timestamps with the correct bins. In practice, each bin contains a list of flows, which is commonly served in *First-In-First-Out* (FIFO) order.

The calendar queue is feasible only if the underlying scheduler guarantees the valid range of timestamp values to be finite at any time. If the finite-range condition holds, then the calendar queue can become a *circular queue*, where the bins are reused as time progresses. In order to prevent timestamps belonging to disjoint intervals from overlapping in the same bin, the range of valid timestamp values must always be a subset of the total range of values that the calendar queue can cover.

2.3 Deficit Round Robin

The *Deficit Round Robin* (DRR) algorithm [7] is one of the most popular instantiations of a WRR scheduler for variable-sized packets, due to its minimal implementation complexity and its efficiency in servicing the flows in proportion to their allocated shares.

Conforming to the WRR paradigm, DRR associates a service share ρ_i with each configured flow i. The service shares translate into *minimum guaranteed service rates* when their sum over all configured flows does not exceed the capacity r of the server:

$$\sum_{i=1}^{V} \rho_i \leq r \tag{1}$$

The bound of eq. (1), where V is the total number of configured flows, guarantees that flow i receives service at a long-term rate that is not lower than ρ_i. In addition to this *minimum-bandwidth guarantee*, the bound always provides a *latency guarantee* to the flow (the provision of the latency guarantee is not possible if the total share allocation exceeds the capacity of the server).

The DRR algorithm divides the activity of the server into *frames*. Throughout this paper, we consider a formulation of the algorithm that uses a *reference timestamp increment* T_Q to express the frame duration in the virtual-time domain. This formulation is functionally equivalent to the definition of DRR originally presented in [7], but is better suited to the description of the WRR enhancements that we present in the following sections.

Within a frame, each configured flow i is entitled to the transmission of a quantum Q_i of information units such that

$$Q_i = \rho_i \cdot T_Q \tag{2}$$

The scheduler visits the backlogged flows only once per frame, and therefore fulfills in a single shot their service expectations for the frame. Each flow i maintains a *timestamp* F_i, which is updated every time a new packet p_i^k of length l_i^k reaches the head of the flow queue:

$$F_i^k = F_i^{k-1} + \frac{l_i^k}{\rho_i} \tag{3}$$

The scheduler keeps servicing the flow as long as its timestamp remains smaller than T_Q. When the timestamp exceeds the reference timestamp increment, the scheduler declares the visit to flow i over: it subtracts T_Q from the timestamp and looks for another backlogged flow to serve. The timestamps carry over the service credits of the backlogged flows to the following frames, allowing the scheduler to distribute service proportionally to the allocated service shares in the long term (*i.e.*, over multiple frames).

When a flow i becomes idle, the scheduler immediately moves to another flow. If flow i becomes backlogged again in a short time, it must wait for the next frame to start in order to receive a new visit from the server. When the flow becomes idle, its timestamp is reset to zero to avoid any loss of service when the flow becomes backlogged again in a future frame. By construction, the timestamp of an idling flow is always smaller than T_Q, so that the timestamp reset never generates extra credits that would otherwise penalize other flows.

The DRR scheduler was implemented in [7] with a single linked list of backlogged flows, visited in FIFO order. The arrangement of the backlogged flows in a single FIFO queue leads to $O(1)$ implementation complexity, provided that the reference timestamp increment T_Q is not smaller than the timestamp increment determined by the maximum-sized packet for the flow with minimum service share:

$$T_Q \geq \frac{L_{max}}{\rho_{min}} \tag{4}$$

If the condition of eq. (4) is not satisfied, the algorithmic complexity of the scheduler explodes with the worst-case number of elementary operations to be executed between consecutive packet transmissions (elementary operations include: flow extraction and insertion in the linked list; timestamp update; comparison of the timestamp with the reference timestamp increment). In fact, the scheduler may have to deny service to the flow for several consecutive frames, until the repeated subtraction of the reference timestamp increment makes the timestamp fall within the $[0, T_Q)$ interval.

The single-FIFO-queue implementation of DRR pays for its minimal implementation complexity in terms of service burstiness and latency. The following bound on the latency of DRR, optimized under the assumption that $T_Q = L_{max}/\rho_{min}$, was provided in [2]:

$$\theta_i^{\mathrm{DRR}} \leq \left[\frac{3(r-\rho_i)}{\rho_{min}} + \frac{\rho_i}{\rho_{min}}\right] \cdot \frac{L_{max}}{r} \tag{5}$$

The latency of DRR is about three times as large as the latency of SCFQ [11], which in turn has by far the worst delay performance among GPS-related schedulers [2]. The reason for the extremely poor delay performance of DRR is in the combination of the single-FIFO-queue implementation of the sorting structure with the credit-accumulation mechanism that is instantiated in the timestamps. In addition to the heavy degradation in latency, a second negative implication of queueing all backlogged flows in a single linked list and exhausting their frame shares in a single visit is the sizable service burstiness that the flows can experience. Typically, a flow i waits for $T_Q - Q_i/r$ time units in between consecutive visits of the server, and then obtains complete control of the server for Q_i/r consecutive time units.

2.4 Surplus Round Robin

A description of *Surplus Round Robin* (SRR) was provided in [8]. The algorithm features the same parameters and variables as DRR, but a different event triggers the update of the timestamp: a flow i receives a new timestamp F_i^k when the transmission of packet p_i^k gets completed, independently of the resulting backlog status of the flow. The end of the frame is always detected *after* the transmission of a packet, and not before: the timestamp carries over to the next frame the *debit* accumulated by the flow during the current frame, instead of the *credit* that is typical of DRR.

An advantage of SRR over DRR is that it does not require to know in advance the length of the head-of-the-queue packet to determine the end of the frame for a flow. Conversely, in order to prevent malicious flows form stealing bandwidth from their competitors, the algorithm cannot reset the timestamp of a flow that becomes idle. The non-null timestamp of an idle flow is eventually obsoleted by the end of the next frame. Ideally, the timestamp should be reset as soon as it becomes obsolete. However, in a scheduler that handles hundreds of thousands or even millions of flows, a prompt reset of all timestamps that can simultaneously become obsolete is practically impossible. We therefore focus throughout the paper on implementations of the SRR algorithm which do not perform any check for obsolescence on the timestamps of the idle flows, and where a newly backlogged flow always resumes its activity with the latest value of the timestamp, however old that value can be. The effect of this assumption is that a newly backlogged flow may have to give up part of its due service the first time it is visited by the server, in consequence of the debit accumulated long time before.

For simplicity of presentation, in the rest of the paper we use the Weighted Round Robin (WRR) name when we allude to DRR or SRR generically, with no explicit reference to their distinguishing features.

3 Soft Bandwidth Segregation

In this section, we present our novel scheme for enforcing bandwidth segregation in a WRR scheduler without modifying its basic structure.

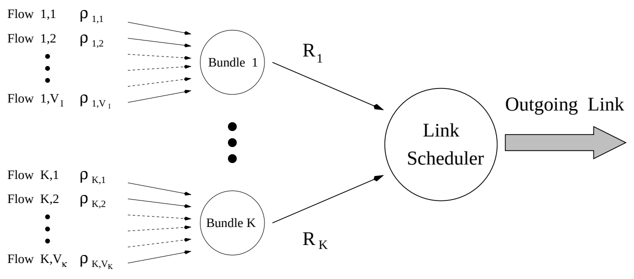


Fig. 1. Reference model for bandwidth segregation.

We show in Fig. 1 the model for bandwidth segregation that we assume as reference. The set of allocated flows is partitioned into K subsets that we call *bundles*. Each bundle I aggregates V_I flows and has an allocated service rate R_I. The logical organization of the scheduler reflects a two-layered hierarchy: it first distributes bandwidth to the bundles, according to their aggregate allocations, and then serves the flows based on their share allocations within the bundles. The scheduler treats each bundle independently of the backlog state of the corresponding flows, as long as at least one of them is backlogged.

We aim at the enforcement of strict bandwidth guarantees for both the flow aggregates and the individual flows within the aggregates (in this context, the flow shares express actual service rates), without trying to support delay guarantees of any sort (frameworks for the provision of stringent delay guarantees in a scheduling hierarchy are already available [3, 4], but they all resort to sophisticated algorithms that considerably increase the complexity of the scheduler). Consistently with the condition of eq. (1) for the provision of per-flow bandwidth guarantees in a WRR scheduler with no bandwidth segregation, the following condition must always hold on the rate allocations of the bundles in the system that we are designing:

$$\sum_{I=1}^{K} R_I \leq r \tag{6}$$

Similarly, the following bound must be satisfied within each bundle I in order to meet the bandwidth requirements of the associated flows:

$$\sum_{i \in I} \rho_i \leq R_I \tag{7}$$

A scheduling solution inspired by the frameworks presented in [3, 4] would introduce a full-fledged (and expensive) scheduling layer to handle the bundles in between the flows and the link server. Generally, the implementation cost of a full-fledged hierarchical scheduler grows linearly with the number of bundles, because each bundle requires a separate instantiation of the basic per-flow scheduler. In our scheme, on the contrary, the layer that enforces the bundle requirements in the scheduling hierarchy is purely virtual, and is superimposed on a *single* instantiation of the basic scheduler. The cost of the structure that handles the individual flows is therefore independent of the number of configured bundles, which leads to substantial savings in the implementation of the scheduling hierarchy.

The addition to the system of a virtual scheduling layer that supports strict bandwidth guarantees for pre-configured flow aggregates relies on a simple modification of the mechanism that maintains the timestamps of the flows in the timestamp-based version of the WRR algorithm (at this point of the discussion, we make no distinction between DRR and SRR). For each configured bundle I, the scheduler maintains, together with the guaranteed aggregate service rate R_I, the sum Φ_I of the service shares of the flows that are currently backlogged in the bundle (we refer to Φ_I as the *cumulative share* of bundle I):

$$\Phi_I = \sum_{i \in \mathcal{B}_I} \rho_i \tag{8}$$

In eq. (8), $\mathcal{B}_I$ is the set of flows of bundle I that are currently backlogged. The idea is to use the ratio between the allocated share of flow i and the cumulative share of bundle I to modulate the guaranteed rate of the bundle in the computation of the timestamp increment associated with packet p_i^k:

$$F_i^k = F_i^{k-1} + \frac{l_i^k}{R_I} \cdot \frac{\Phi_I}{\rho_i} \tag{9}$$

In order to verify that the timestamp assignment rule of eq. (9) actually enforces the bandwidth guarantees of the bundles, we compute the amount of service that bundle I may expect to receive during a frame. The computation is based on the following two assumptions: (i) the cumulative share of the bundle remains unchanged during the whole frame, independently of the backlog dynamics of the corresponding flows; and (ii) the set of flows that can access the server during the frame includes only the flows that are backlogged at the beginning of the frame (if some flows in the bundle become backlogged after the frame has started, they must wait until the beginning of a new frame before they can access the server).

If we apply the rule of eq. (9) over all the services received by flow i of bundle I during the frame, the reference per-frame timestamp increment that we obtain for the flow is:

$$T_Q = \frac{Q_i}{R_I} \cdot \frac{\Phi_I}{\rho_i} \tag{10}$$

Then, by aggregating the service quanta of all the flows in bundle I, we obtain the service quantum Q_I of the bundle:

$$Q_I = \sum_{i \in \mathcal{B}_I} Q_i = \frac{\sum_{i \in \mathcal{B}_I} \rho_i}{\Phi_I} \cdot R_I \cdot T_Q = R_I \cdot T_Q \tag{11}$$

The expression of Q_I in eq. (11) is identical to the expression of the flow quantum Q_i in eq. (2), and therefore proves that the timestamp-updating rule of eq. (9) preserves the bandwidth guarantees of bundle I, independently of the composition of the set of flows that are backlogged in the bundle at the beginning of the frame.

Sticking on the assumption that the cumulative share of bundle I does not change during the frame, we can also show that the timestamp-updating rule of eq. (9) preserves the service proportions for any two flows i, j of bundle I that never become idle during the frame:

$$\frac{Q_i}{Q_j} = \frac{\rho_i \cdot \frac{R_I}{\Phi_I} \cdot T_Q}{\rho_j \cdot \frac{R_I}{\Phi_I} \cdot T_Q} = \frac{\rho_i}{\rho_j} \tag{12}$$

In order to specify the details of the WRR algorithm with bandwidth segregation, we must first discuss the assumptions that we have made to obtain the results of eqs. (11) and (12), and then evaluate their algorithmic implications.

The use of a constant value of cumulative share Φ_I in all the timestamp increments that the scheduler computes during a frame provides a common reference for consistently distributing service to the flows of bundle I. Identical purpose has the exclusion from the frame of the flows that become backlogged only after the frame has started. The timestamp increment is the charge that the system imposes on a flow for the transmission of the related packet. The cost of the transmission depends on the bandwidth that is available within the bundle at the time it is executed. In order to make the timestamp increment consistent with the cost of the bandwidth resource within the bundle, it must be computed when the resource is used, *i.e.*, upon the transmission of the corresponding packet. If the scheduler computes the increment in advance, the state of the bundle (and therefore the actual cost of the bandwidth resource) can undergo radical changes before the transmission of the packet occurs, thus making the charging mechanism inconsistent with the distribution of bandwidth.

Within the pair of WRR algorithms that we are considering, SRR is the one that best fits the requirement for consistency between transmissions and timestamp increments, because it uses the length of the just transmitted packet to update the timestamp and determine the in-frame status of the corresponding flow. In DRR, on the contrary, the scheduler performs the timestamp update and the in-frame status check using the length of the new head-of-the-queue packet, possibly long before it is actually transmitted. When the DRR server finally delivers the packet, the cumulative share of the bundle, and therefore the cost of bandwidth within the bundle, may have changed considerably since the latest timestamp update.

Introducing the mechanism for bandwidth segregation in SRR is straightforward. In addition to the minimum-bandwidth guarantee R_I and the cumulative share Φ_I, each bundle I maintains a *running share* ϕ_I and a *start flag* σ_I. The running share keeps track of the sum of the service shares of the backlogged flows in the bundle:

$$\phi_I(t) = \sum_{i \in \mathcal{B}_I(t)} \rho_i \quad \forall t \tag{13}$$

The running share is updated every time a flow of the bundle changes its backlog state. In general, the updates of the running share ϕ_I do not translate into immediate updates of the cumulative share Φ_I. In fact, the scheduler updates the cumulative share of the bundle only upon detection of mismatching values in the start flag σ_I and in a global single-bit frame counter $FRMCNT$ that the scheduler toggles at every frame boundary (the scheduler compares σ_I and $FRMCNT$ every time it serves a flow of bundle I). A difference in the two bits triggers the update of the cumulative share to be used in the future timestamp computations ($\Phi_I \leftarrow \phi_I$) and toggles the start flag of the bundle ($\sigma_I \leftarrow FRMCNT$). If, instead, the two bits are already equal, the service just completed is certainly not the first one that the bundle receives during the current frame, and no action must be taken on the bundle parameters. When the first flow of a bundle becomes backlogged, the start flag is set to the complement of $FRMCNT$.

In order to identify the end of a frame, each flow i maintains a *frame flag* FF_i. The frame flag of flow i is set to the complement of $FRMCNT$ whenever the flow is queued to the tail of the list of backlogged flows. When the scheduler finds a frame flag that does not match the frame counter, it declares the start of a new frame and toggles the frame counter. The sequence of operations to be executed after completing the transmission of a packet and processing the corresponding flow is summarized in the pseudo-code of Fig. 2.

We illustrate in a simple simulation scenario the behavior of the SRR scheduler with bandwidth segregation. We consider a packet multiplexer with capacity r and three configured bundles (I, J, and K). Bundle I is allocated 60% of the server capacity, and contains 52 flows: the first 50 flows ($i_1, \ldots, i_{50}$) are each allocated 0.2% of the capacity of the server; the service share of flow i_{51} is $\rho_{i_{51}} = 0.2\,r$; finally, the service share of flow i_{52} is $\rho_{i_{52}} = 0.3\,r$. Flows $i_1, \ldots, i_{50}$ are strictly regulated, *i.e.*, their packet arrival rate at the multiplexer never exceeds their guaranteed service share. These flows typically have negligible backlog whenever the server has bandwidth available in excess of their guaranteed shares. As a consequence, we expect each of them to switch backlog status quite often. Flow i_{51} is unregulated: the packet arrival rate of the flow is far above its allocated service share. This behavior induces a permanent backlog in the packet queue of flow i_{51}. Finally, flow i_{52} always remains inactive, with no packets reaching the multiplexer. Bundle J is allocated 20% of the capacity of the server, and contains a single, unregulated flow j_1 (similarly to i_{51}, the backlog in the queue of flow j_1 is permanent, independently of the service sequence generated by the scheduler). Bundle K is configured exactly the same way as bundle J: it is allocated 20% of the capacity of the server, and contains a single,

```
Identify flow $i$ currently at the head of the linked list
Identify bundle $I$ of flow $i$
**if** $(FF_i \neq FRMCNT)$
    $FRMCNT \leftarrow \neg FRMCNT$
Prepare head-of-the-queue packet $p_i^k$ for transmission
**if** $(\sigma_I \neq FRMCNT)$
    $\Phi_I \leftarrow \phi_I$
$\sigma_I \leftarrow FRMCNT$
$F_i^k \leftarrow F_i^{k-1} + \frac{l_i^k}{R_I} \cdot \frac{\Phi_I}{\rho_i}$
**if** $(F_i^k \geq T_Q)$ /* Frame over for flow $i$ */
    $F_i^k \leftarrow F_i^k - T_Q$
    $FF_i \leftarrow \neg FRMCNT$
    Extract flow $i$ from head of linked list
    **if** (Flow $i$ is still backlogged)
        Append flow $i$ to tail of linked list
    **else** /* Flow $i$ is getting idle */
        $\phi_I \leftarrow \phi_I - \rho_i$
**else if** (Flow $i$ is getting idle)
    Extract flow $i$ from head of linked list
    $\phi_I \leftarrow \phi_I - \rho_i$
```

Fig. 2. Pseudo-code of SRR with bandwidth segregation.

unregulated flow k_1. The inactivity of flow i_{52} makes 30% of the capacity of the server available to the flows that are active. The objective of the simulation setup is showing that the association of the active flows with three different bundles has relevant impact on the distribution of bandwidth that is in excess of the nominal guarantees.

We execute two simulation runs under the same pattern of packet arrivals. In the first run, we remove the bundles from the system, and schedule the 53 active flows based on their allocated shares $\rho_{i_1}, \ldots, \rho_{i_{51}}, \rho_{j_1}$, and ρ_{k_1}. We restore the bundles in the second run. We measure in both cases the bandwidth received individually by flows i_{51}, j_1, and k_1, and cumulatively by the aggregate of flows $i_1, \ldots, i_{50}$, and report the results in Table 1. When no bundles are superimposed, we expect the excess bandwidth made available by the inactivity of flow i_{52} to be evenly shared by flows i_{51}, j_1 and k_1 (flows $i_1, \ldots, i_{50}$ have no access to the excess bandwidth because they have no extra supply of packets above their nominal bandwidth allocations). When the bundles are overlaid, on the contrary, i_{51} is the only flow that is entitled to receive extra services, as long as it remains backlogged.

The experimental results substantially match the ideal shares that the scheduler should distribute, giving evidence to the effectiveness of the bundles in segregating bandwidth.

Flow	Throughput				
	Nominal	Without Bundles		With Bundles	
	Allocation	Expected	Observed	Expected	Observed
$i_1 + \ldots + i_{50}$	10.00	10.00	9.98	10.00	9.99
i_{51}	20.00	30.00	30.00	50.00	49.95
i_{52}	30.00	0.00	0.00	0.00	0.00
j_1	20.00	30.00	30.01	20.00	20.03
k_1	20.00	30.00	30.01	20.00	20.03

Table 1. Bandwidth segregation in SRR. Throughput is expressed in percentages of the server capacity.

4 Reducing the Service Burstiness

In the previous section, we have identified SRR as the algorithm that successfully enforces bandwidth segregation with no substantial impact on the implementation complexity. However, the single-FIFO-queue implementation of the scheduler still induces extremely poor performance in terms of latency and service burstiness. The scheduler visits each flow only once per frame: in order to maintain the bandwidth proportions defined by the allocated service shares, the scheduler must fulfill the entire per-frame service expectation of the flow in a single shot. If a flow has a large number of in-frame packets waiting for service, the server transmits them back-to-back until the flow timestamp exceeds the reference timestamp increment T_Q.

It is natural to expect a reduction in latency and burstiness if the server interleaves the transmission of packets of different flows within the same frame. This interleaving of packet transmissions is possible only if flows with more than one packet to transmit can reach the head of the list of backlogged flows multiple times during the frame. A *single* FIFO queue of flows obviously collides with this requirement, because a flow that is extracted from the head of the list is automatically forced to enter the next frame. Thus, the only way to distribute smoother services is increasing the number of queues of backlogged flows that the scheduler maintains.

The calendar queue that we have reviewed in Section 2.2 is an array of linked lists, where flows get queued based on the values of the associated timestamps. We resort to this technique, which relates the service pattern generated by the scheduler to the progress in the amount of information that the flows have already transmitted during the frame, to reduce the service burstiness of our scheduler.

The calendar queue that we use to implement SRR, shown in Fig. 3, is divided in two logical segments of equal size. The *in-frame* segment contains the backlogged flows that can still be serviced during the current frame, while the *out-of-frame* segment contains the flows with exhausted service share and the newly backlogged flows of the current frame. At every end of a frame, the

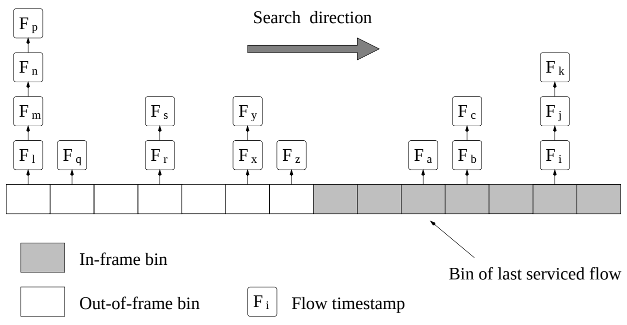


Fig. 3. Calendar queue for the implementation of SRR.

in-frame and out-of-frame segments swap their position in the array of bins. In a calendar queue with N_b bins in each segment, the bin associated with the timestamp F_i^k of flow i has the following offset b_i within its segment:

$$b_i = \left\lfloor N_b \cdot \frac{F_i^k}{T_Q} \right\rfloor \tag{14}$$

When a flow becomes backlogged, it is queued to the tail of the bin corresponding to its latest timestamp in the out-of-frame segment of the calendar queue. When a flow is serviced, it is first extracted from the head of its current bin. Then the scheduler updates its timestamp, and checks whether the new value exceeds T_Q. If this is the case, the scheduler subtracts T_Q from the timestamp and, if the flow is still backlogged, appends it to the proper bin in the out-of-frame segment of the calendar queue. If, instead, the new timestamp does not exceed T_Q, the scheduler appends the flow to a bin of the in-frame segment.

Every time the transmission of a packet is completed, the scheduler searches for the next flow to serve, starting from the bin of the just serviced one. If the first non-empty bin is found in the out-of-frame segment, the scheduler toggles the frame counter $FRMCNT$, which declares the beginning of a new frame. There is no longer need to maintain per-flow frame flags, as in the single-FIFO-queue implementation of SRR, because the scheduler detects the start of a new frame directly from the position of the first non-empty bin, without having to retrieve the information from the flow selected for service.

In spite of the evident qualitative improvement that the interleaved packet transmissions introduce in the system, the worst-case indices that typically characterize a packet scheduler (latency, *worst-case fairness index* [3], and *service fairness index* [11]) barely obtain any benefit from the calendar-queue implementation of SRR. The reason for this counter-intuitive outcome is that all those indices are dominated by the worst possible state that a flow can find in the system when it is newly backlogged. Since the worst-case state of the system is practically independent of the adopted implementation, the indices look very similar with the single FIFO queue and the calendar queue. We prefer not to proceed with a detailed derivation of the indices, and only point out that they are all dominated by the term $2L_{max}/\rho_{min}$, which is the time that the server takes to transmit a two-frames worth of backlog for all the flows configured in the system, assuming that each of them is allocated the minimum share ρ_{min}. The considerable benefits of the calendar-queue implementation of SRR can be appreciated as soon as some of the configured flows have service share greater than ρ_{min}, a situation that none of the common worst-case indices can capture.

We use a simple simulation experiment to show the performance of the calendar-queue implementation of SRR under less extreme traffic conditions. We configure 53 unregulated flows, divided in three distinct classes. All configured flows belong to the same bundle, so that the higher level of the scheduling hierarchy remains transparent. The first class includes 50 flows ($i_1, \ldots, i_{50}$), and each flow in the class is allocated 0.4% of the capacity of the server. The second class consists of two flows (i_{51} and i_{52}), with a bandwidth allocation of $0.2\,r$ for each of them. Flow i_{53} is the only member of the third class, and has a bandwidth allocation of $0.4\,r$. For simplicity, all packets reaching the system have the same size. We set the minimum service share ρ_{min} that the scheduler can support to three different values in three distinct simulation runs: $\rho_{min}^{(1)} = \rho_{i_1}$, $\rho_{min}^{(2)} = 0.1\,\rho_{i_1}$, and $\rho_{min}^{(3)} = 0.01\,\rho_{i_1}$. The reference timestamp increment T_Q, which determines the duration of a frame, increases as the ratio between the service shares of the configured flows and ρ_{min} increases. We implement the SRR scheduler with the single-FIFO-queue technique (SQ), and with two calendar queues (CQ1 and CQ32), having respectively 1 and 32 bins per segment (for a total of 2 and 64 bins per calendar queue). In a server that guarantees a fixed number of packet transmissions to a flow that remains continuously backlogged, the maximum time elapsing between the transmissions of two consecutive packets of a flow is a clear indicator of the service burstiness of the scheduler.

Flow	$\rho_{min}^{(1)}$			$\rho_{min}^{(2)}$			$\rho_{min}^{(3)}$		
	SQ	CQ1	CQ32	SQ	CQ1	CQ32	SQ	CQ1	CQ32
i_1	250	250	250	2491	2023	1023	2491	2023	299
i_{53}	153	53	53	1501	53	53	1501	53	53

Table 2. Maximum inter-departure times for flows i_1 ($\rho_{i_1} = 0.004\,r$) and i_{53} ($\rho_{i_{53}} = 0.4\,r$) for different values of minimum configurable service share ($\rho_{min}^{(1)} = \rho_{i_1}$, $\rho_{min}^{(2)} = 0.1\,\rho_{i_1}$, and $\rho_{min}^{(3)} = 0.01\,\rho_{i_1}$). The inter-departure times are normalized to the transmission time of a packet.

In Table 2, we report the maximum inter-departure times observed for flows i_1 ($\rho_{i_1} = 0.004\,r$) and i_{53} ($\rho_{i_{53}} = 0.4\,r$) in a sequence of delivered packets that is long enough to cover the duration of a frame. In the first scenario, with $\rho_{min}^{(1)} = \rho_{i_1}$, there is no substantial difference between the single-FIFO-queue implementation and the two calendar queues, because flows $i_1, \ldots, i_{50}$ are entitled to the transmission of only one packet per frame. As the number of packets transmitted in a frame by the flows with lowest share grows, the benefit of the calendar-queue implementation becomes more and more evident.

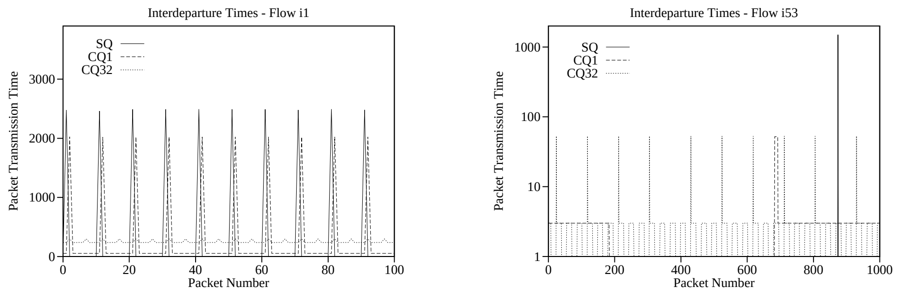


Fig. 4. Inter-departure times observed for flows i_1 ($\rho_{i_1} = 0.004\,r$) and i_{53} ($\rho_{i_{53}} = 0.4\,r$), normalized to the transmission time of a packet. The plot for flow i_{53} reports the inter-departure times in logarithmic scale, in order to distinguish the behavior of the three implementations at the lower end of the covered range.

At first glance, the performance of the two calendar-queue implementations is very similar, especially for the flow with highest service share (the detected maximum inter-departure times reported in Table 2 for flow i_{53} are always the same with CQ1 and CQ32). In order to better appreciate the effect of increasing the number of bins in the calendar queue, it is necessary to look with more detail at the sequence of inter-departure times produced by the scheduler. In Fig. 4, we trace the inter-departure times that the SQ, CQ1, and CQ32 implementations of SRR produce for flow i_1 (i_{53}) in the transmission of 100 (1000) consecutive packets, in the case where $\rho_{min} = \rho_{min}^{(2)} = 0.1\,\rho_{i_1}$. We immediately observe that a higher number of bins determines much shorter sequences of consecutive packets that are spaced by the maximum inter-departure time, which testifies lower burstiness.

5 Concluding Remarks

In this paper, we have presented two enhancements of WRR schedulers for providing bandwidth guarantees in IP networks. In our first enhancement, we superimpose a "soft" hierarchical structure to a WRR scheduler, which allows to segregate bandwidth among bundles of flows, in addition to providing bandwidth guarantees to the individual flows. The mechanism has minimal complexity, since it is entirely based on redefining the way timestamps are computed. In our second enhancement, we achieve a considerable reduction in service burstiness by implementing the WRR scheduler with a modified calendar queue. The improvement is substantial even with few sorting bins in the calendar queue; thus, the required additional complexity is rather small.

Both enhancements are useful mechanisms to meet the increasingly sophisticated scheduling demands in networks supporting differentiated services, while keeping the complexity of the schedulers to a minimum.

References

[1] D. Stiliadis and A. Varma, "Design and Analysis of Frame-based Fair Queueing: A New Traffic Scheduling Algorithm for Packet-Switched Networks," *Proceedings of ACM SIGMETRICS '96*, pp. 104–115, May 1996.

[2] D. Stiliadis and A. Varma, "Latency-rate Servers: A General Model for Analysis of Traffic Scheduling Algorithms," *IEEE INFOCOM '96*, pp. 111–119, March 1996.

[3] J. C. R. Bennett and H. Zhang, "Hierarchical Packet Fair Queueing Algorithms," *Proceedings of ACM SIGCOMM '96*, pp. 143–156, August 1996.

[4] I. Stoica, H. Zhang, and T. S. E. Ng, "A Hierarchical Fair Service Curve Algorithm for Link-Sharing, Real-Time and Priority Services," *Proceedings of ACM SIGCOMM '97*, September 1997.

[5] D. C. Stephens, J. C. R. Bennett, and H. Zhang, "Implementing Scheduling Algorithms in High-Speed Networks," *IEEE Journal on Selected Areas in Communications*, Vol. 17, No. 6, June 1999, pp. 1145-1158.

[6] M. Katevenis, S. Sidiropoulos, and C. Courcoubetis, "Weighted Round Robin Cell Multiplexing in a General-Purpose ATM Switch," *IEEE Journal on Selected Areas in Communications*, vol. 9, pp. 1265–79, October 1991.

[7] M. Shreedhar and G. Varghese, "Efficient Fair Queueing Using Deficit Round Robin," *IEEE/ACM Transactions on Networking*, vol. 4, no. 3, pp. 375–385, June 1996.

[8] H. Adiseshu, G. Parulkar, and G. Varghese, "A Reliable and Scalable Striping Protocol," *Proceedings of ACM SIGCOMM '96*, August 1996.

[9] A. K. Parekh and R. G. Gallager, "A Generalized Processor Sharing Approach to Flow Control in Integrated Services Networks: The Single-Node Case," *IEEE/ACM Transactions on Networking*, pp. 344–357, June 1993.

[10] L. Zhang, "Virtual Clock: A New Traffic Control Algorithm for Packet Switching," *ACM Transactions on Computing Systems*, pp. 101–124, May 1991.

[11] S. J. Golestani, "A Self-Clocked Fair Queueing Scheme for Broadband Applications," *Proceedings of IEEE INFOCOM '94*, pp. 636–646, April 1994.

[12] P. Goyal, H. M. Vin, and H. Chen, "Start-time Fair Queueing: A Scheduling Algorithm for Integrated Services," *ACM SIGCOMM '96*, pp. 157–168, August 1996.

[13] J. L. Rexford, A. G. Greenberg, and F. G. Bonomi, "Hardware-Efficient Fair Queueing Architectures for High-Speed networks," *Proceedings of IEEE INFOCOM '96*, pp. 638–646, March 1996.

[14] J. L. Rexford, A. G. Greenberg, F. G. Bonomi, and A. Wong, "Scalable Architectures for Integrated Traffic Shaping and Link Scheduling in High-Speed ATM Switches," *IEEE Journal on Selected Areas in Communications*, vol. 15, no. 5, pp. 938–950, June 1997.

[15] H. J. Chao and J. S. Hong, "Design of an ATM Shaping Multiplexer with Guaranteed Output Burstiness," *Computer Systems Science and Engineering*, vol. 12, no. 2, March 1997.
[16] F. M. Chiussi, A. Francini, and J. G. Kneuer, "Implementing Fair Queueing in ATM Switches – Part 2: The Logarithmic Calendar Queue," *Proceedings of IEEE GLOBECOM '97*, pp. 519–525, November 1997.

Router Architectures Exploiting Input-Queued Cell-Based Switching Fabrics*

Marco Ajmone Marsan, Andrea Bianco, Paolo Giaccone, Emilio Leonardi, and Fabio Neri

Dipartimento di Elettronica, Politecnico di Torino, Torino 10129, Italy,
{ajmone,bianco,giaccone,leonardi,neri}@polito.it,
http://www1.tlc.polito.it/

Abstract. Input queued and combined input/output queued switching architectures must be controlled by a scheduling algorithm, which solves contention in the transfer of data units to switch outputs. We consider the case of packet switches (or routers), i.e., devices operating on variable-size data units at their interfaces, assuming that they internally operate on fixed-size data units, and we propose novel extensions of known scheduling algorithms for input queued and combined input/output queued architectures. We show by simulation that such architectures can provide performance advantages over traditional output queued architectures.

1 Introduction

The effort in the design of ATM switches brought on the market efficient chipsets currently used as building blocks for high-performance Internet routers; as a consequence, for many advanced Internet routers the switching fabric internally operates on fixed-size data units (named *cells*), and input IP datagrams are internally segmented into ATM-like cells that are transferred to output interfaces, where they are reassembled into variable-size IP datagrams.

Output queuing (OQ) architectures have been traditionally used to build switches and routers since they provide optimal performance. The main disadvantage of OQ architectures is that both the internal interconnect (i.e., the switching fabric) and output queues in line cards must operate at a speed equal to the sum of the rates of all input lines. In other words, an internal speed-up is needed to transfer data units to output interfaces. In applications where the number of ports is large or the line rate is high, this makes OQ impractical.

Input-queued (IQ) architectures [1] have recently come back into the arena as packet switching engines, and they are currently considered by many designers as the best solution when the line speed is pushed to technological limits. IQ schemes permit all the components of the switch (input interfaces, switching fabric, output interfaces) to operate at a speed which is compatible with the

* This work was supported by the Italian Ministry for University and Research through the MQOS Project

data rate of input and output lines. In other words, no internal speed-up is required.

A major issue in the design of IQ switches is that the access to the switching fabric must be controlled by some form of scheduling algorithm. Note that we use the term "scheduling algorithm" in this paper only for switching matrix schedulers, that decide which input port is enabled to transmit in a IQ switch; they avoid blocking and solve contention within the switching fabric. The same term "scheduling algorithm" is sometimes used to describe flow-level schedulers, that decide which cell flows must be served in accordance to QoS requirements. Several scheduling algorithms for IQ cell switches were proposed and compared in the literature [2,3,4,5,6]. They provide performance very close to the more hardware-demanding OQ architecture. We consider some of these proposals, and modify them to deal with variable-size packets: we constrain the scheduling algorithm to deliver contiguously all the cells deriving from the segmentation of the same packet. In other words, variable-size packets are transformed into "trains of cells", and the transfer of the cells belonging to the same train is scheduled in such a way that they remain contiguous in the delivery to the output card, i.e., they are not interleaved with the cells of another train.

Scheduling algorithms for IQ architectures are always relatively demanding in terms of computing power and control bandwidth. It has been shown [7] that this complexity can be partly reduced when the switching fabric and the output memory bandwidth have a moderate speed-up with respect to the data rate of input/output lines. A speed-up two, independent of the number of switch ports, can be shown [8,9,10] to be sufficient to provide the same performance of an OQ architecture. As soon as the switch takes advantage of an internal speed-up, buffering is required at output ports; we use the term combined input/output queueing (CIOQ) to describe this architecture.

In this paper we study IQ and CIOQ switching architectures, taking OQ switches as a reference for comparisons. We concentrate on switch architectures that internally operate in a "synchronous" fashion, i.e., in which switching decision are taken at equally spaced time instants, but in general accept at their inputs variable-size data units. We devote some attention to CIOQ architectures with a speed-up equal to two, where input buffers are organized into a single FIFO queue, and the scheduling algorithm is particularly simple. A comparison of this setup with IQ architectures is presented.

2 Logical Architecture

We describe now the logical structure of cell and packet switches, based on IQ, OQ and CIOQ switching architectures.

2.1 Cell Switches

We consider a switch with N inputs and N outputs. We also assume for simplicity that all input and output lines run at the same speed.

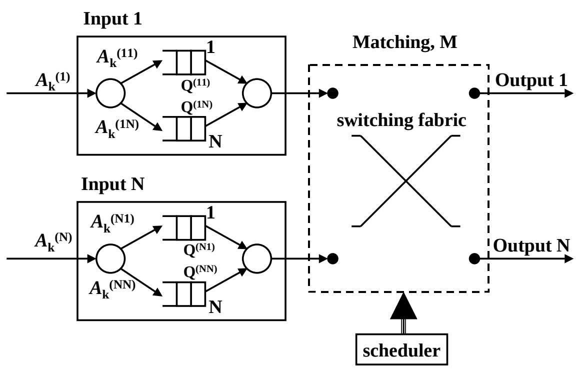


Fig. 1. Logical structure of an input-queued cell switch

The switch operates on fixed-size data units, named *cells*. Actually, we refer to any switch that takes switching decision at equally-spaced time instants. The distance between two switching decisions is called *slot*.

Input-Queued Cell Switches Fig. 1 shows the logical structure for an input-queued **cell** switch. Cells are stored at input interfaces. Each input manages one queue for each output, hence a total of $N \times N = N^2$ queues are present. This queue separation permits to avoid performance degradations due to head-of-the-line blocking [11], and is called Virtual Output Queuing (VOQ) or Destination Queuing [1,7].

Cells arrive at input i, $1 \leq i \leq N$, according to a discrete-time random process $A_i(k)$. At most one cell per slot arrives at each input, i.e., the data rate on input lines is no more than 1 cell per slot. When a cell with destination j arrives at input i, it is stored in the FIFO queue Q_{ij}. The number of cells in Q_{ij} at time k is denoted by $L_{ij}(k)$. These FIFO queues have limited capacity: each queue can store at most L cells.

The switching fabric is non-blocking and memoryless, and introduces no delay: at most one cell can be removed from each input and at most one cell can be transferred to each output in every slot. Since the speed at which cells are fed into output interfaces is equal to the speed at which cells are fed to input interfaces, we have a speed-up factor (see Section 2.2) equal to 1. The scheduling algorithm decides which cells can be transferred from the inputs to the outputs of the switch in every slot.

Output-Queued Cell Switches The output-queued cell switch needs no input buffers because the switching fabric has enough capacity to transfer to the desired output all the cells received in one time slot. In the worst case (i.e., when a cell arrives at each input, and all cells are directed to the same output), this means that the bandwidth towards each output must be equal to the sum of the bandwidths available on all input lines. Thus, we say that the switching fabric must have a speed-up factor equal to N.

At each output, cells are stored in a single FIFO queue. For a fair comparison with input-queued switches, we assume that the total amount of buffer space is kept constant, i.e., that the FIFO output queue can store $N \times L$ cells. This assumption gives some advantage to the OQ schemes, which can exploit some degree of buffer sharing.

Combined Input/Output-Queued Cell Switches CIOQ architectures are a compromise between IQ and OQ, for which some degree of speed-up is available, but not as much as for OQ. These architectures require buffering at both the input and output line interfaces.

The buffer space at each input can be organized in either one FIFO queue, or several queues, like in the case of VOQ. We call S_{in} the number of cells per slot that can be read from the queue(s) at each input, and S_{out} the number of cells per slot that can be written into the output queue at each output.

Several CIOQ architectures were proposed in the literature, with different speed-up definition. We adopt the following definition [7]: $S_{in} = S_{out} = S$. Thus, up to S cells can be read from each input and written to each output of the switch in one time slot. Of course, when $S = 1$, we have an IQ architecture.

It is well-known that the maximum normalized throughput achievable in an IQ switch using one FIFO queue per input port, with uniform traffic and an infinite number of ports, is limited to 0.586 by the head-of-the-line blocking phenomenon [11]. One can argue that, by executing the scheduling algorithm twice in each slot, thereby using a speed-up $S = 2$, the maximum throughput becomes 100%, as it is for OQ architectures. This simple statement holds only for uniform traffic, and does not provide guarantees related to delays.

It has been shown that a CIOQ architecture with VOQ can mimic the OQ behavior if a speed-up $S = 2$ is available, and if a (non trivial) scheduling algorithm called Home Territory Algorithm is implemented [7].

In this paper, with the aim of being fair in the comparison of IQ, OQ and CIOQ architectures, we accept high scheduling complexity for IQ, high hardware complexity (i.e., speed-up N) for OQ, and intermediate scheduling and hardware complexity for CIOQ. We therefore limit our attention to CIOQ switches with limited speed-up increase $S = 2$, and very simple FIFO scheduling.

Although a more detailed discussion of scheduling algorithms for IQ architectures will be provided in Section 3, we anticipate here the description of the simple algorithm that we consider in the CIOQ case. We assume that each input and each output of the CIOQ switch maintain a single FIFO queue, and that the following scheduling algorithm is executed twice in each slot.

At every execution of the algorithm, inputs are cyclically scanned, starting from a different input every time, selected by a round-robin scan. In this scan each input attempts to transfer the cell at the head of its input FIFO queue. If the corresponding output was not already engaged by a preceding input in the round-robin scan, the transfer is enabled, otherwise it is deferred to the next

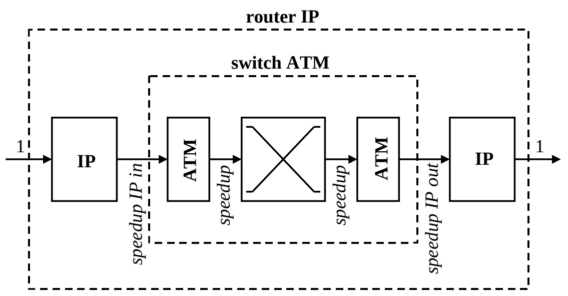


Fig. 2. Speed-up definitions for the packet switch built around a cell switch

execution of the algorithm. Since the algorithm is executed twice in each slot time, at most the first two cells are removed from input queues, and at most two cells are delivered to each output queue in each slot. We call this scheduling algorithm FIFO–2.

2.2 Packet Switches (or Routers)

We use as a reference model the case of a high-performance IP router built around an ATM cell-switch. The IP protocol sits on top of any data-link protocol at the input and at the output of the router. Input IP datagrams are segmented into ATM cells, that will be transferred to output ports by a high-performing ATM switching fabric. Cells delivered to an output port are reassembled into the corresponding IP datagram.

Fig. 2 emphasizes the possible speed variations inside an IP router incorporating an ATM switching fabric. We take as a reference the bit rates on the input and output lines, which we assume to be the same and equal to 1.

- SPEEDUP-IP-IN is the speed at which cells are transferred from the input IP module to the input of the ATM switch. SPEEDUP-IP-IN= 1 means that, if an input IP datagram is segmented in k cells, all these cells are sequentially transferred to the cell-switch input in k time slots. Note that this takes into account segmentation overheads and partial filling of the last cell.
- SPEEDUP is the number of cells per slot that can be read from the inputs of the cell-switch. Due to the definition of the speed-up S for cell-switches, this is also the number of cells per slot that can be written onto a switch output.
- SPEEDUP-IP-OUT is the speed at which cells are transferred from the output of the cell-switch to the output IP module.

Input and output IP modules in Fig. 2 consist of queues for variable-size packets, and operate in store-and-forward mode; they comprise segmentation and reassembly functions. ATM modules comprise cell queues at the input and/or at the output of the switch.

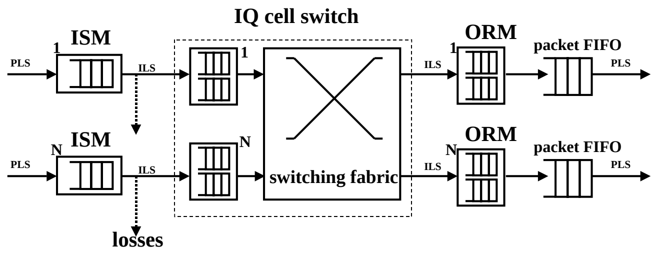


Fig. 3. Logical architecture for an IQ packet switch

IQ Packet Switches The logical architecture for an IQ packet switch is shown in Fig. 3. At each input an Input Segmentation Module (ISM) segments the incoming packet into cells. PLS is the external packet line speed. Since the ISM operates in store-and-forward mode, it must be equipped with enough memory to store a maximum-size packet, and the segmentation process starts only after the complete reception of the packet.

The cells resulting from the segmentation are transferred to the cell-switch input at a speed (called ILS) equal to the line speed PLS incremented to account for segmentation overheads. The capacity of input queues at the cell-switch is limited to L, hence losses can occur. We assume that the entire packet is discarded if the input queue of the cell-switch does not have enough free space to store all the cells deriving from the segmentation of the packet *when the first of these cells hits the queue.* This is of course a pessimistic assumption, but has the advantage of ease of implementation, and of avoiding the transmission of incomplete packet fractions through the switch.

The cell-based switching fabric transfers cells from input to output queues, according to a scheduling algorithm. These cells are delivered to the Output Reassembly Module (ORM) at speed ILS. Here packets (i.e., IP datagrams) are reassembled. In general, cells belonging to different packets can be interleaved at the same output, hence more than one reassembly machine can be active in the same ORM. However, at most one cell reaches each ORM in a slot time, hence at most one packet is completed at each ORM in a slot.

Once a packet is complete, it is logically added to an output packet queue, called *packet FIFO* in the figure, from which packets are sequentially transmitted onto the output line. No internal speedup is required to support ISM and ORM, but for compensating internal overheads.

Optimization of IQ Packet Switches In the case of IQ packet switches, it is possible to further simplify the structure of the switch, and to improve its performance, by enforcing additional constraints on the scheduling algorithm. Indeed, the cells belonging to the same packet are contiguous in the input queue of the internal cell-switch. As we shall see in Section 3, it is possible to modify some well-known scheduling algorithms in such a way that, once the transfer through

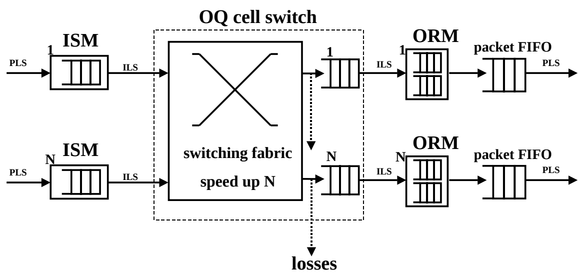


Fig. 4. Logical architecture for an OQ packet switch

the switching fabric of the first cell of a packet has started towards the corresponding output port, no cells belonging to other packets can be transferred to that output. We call this class of scheduling algorithms *packet-mode scheduling.* Note that in OQ switches the interleaving of cells in output queues cannot be avoided.

If packet-mode scheduling is adopted in IQ packet switches, since cells belonging to the same packet are kept contiguous also in the output queue, the logical architecture can be simplified: both the ORM module and the output packet FIFO can be removed with respect to Fig. 3. The removal of the packet FIFO can be achieved only if no format conversion is required to transmit the packet on the output link. Since each module operates in store-and-forward mode, hence introduces a delay equal to the packet size, the delays through the switch are reduced by twice the packet duration.

These observations show that important additional advantages with respect to cell switches are exhibited by IQ architectures with respect to OQ architectures (see next paragraph) for the implementation of packet switches.

OQ Packet Switches Fig. 4 shows the logical architecture of an OQ packet switch. Packets arrive at input ports where they are segmented by ISM modules, similarly to what happens in IQ routers. The cells obtained with the segmentation are sent to the cell-switch at speed ILS, and are immediately transferred to the output queues of the cell-switch, thanks to a speed-up equal to N in the switching fabric. Losses may occur at output queues, whose capacity is limited to $N \times L$ for each queue, since we are assuming the same buffering capacity for OQ and IQ switches.

From the output queues of the cell-switch, cells are delivered at speed ILS to an ORM module for reassembly. Once a packet is completed, it is queued in a packet FIFO queue similar to what was seen for the IQ case.

An important difference between OQ packet switches and IQ packet switches resides in the way of handling losses. Cells belonging to different packets can be interleaved in output queues. When a cell at the output of the switching fabric does not find room in the output queue of the cell-switch, it must be discarded.

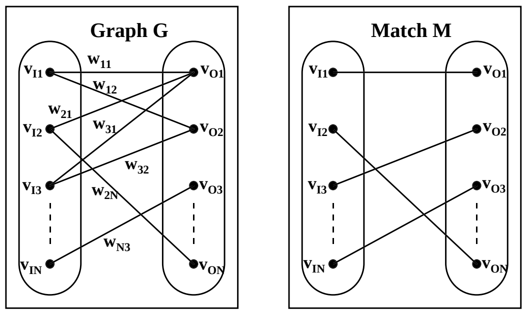


Fig. 5. Bipartite graph description of the scheduling problem

The other cells belonging to the same packet of the discarded cell may be in the output queue, or already in the ORM. We assume to be unable to identify and discard the other cells in the output queue: those cells will be discarded by the ORM module, which has knowledge of the existence of the packet. This means that cells belonging to packets that suffered partial losses unnecessarily use system resources.

CIOQ Packet Switches CIOQ packet switches are built around a CIOQ cell switch. For the sake of conciseness, we do not show the logical architecture in this case, but it can be easily obtained by combining Figs. 3 and 4. The only difference is that we do not consider VOQ in the CIOQ case, hence only one FIFO queue is available at each input (and at each output) of the cell switch. Losses may occur at both input and output interfaces.

3 Cell and Packet Scheduling Algorithms in IQ Switches

3.1 Problem Definition

In the technical literature, scheduling algorithms in IQ cell switch architectures are described as solutions of the matching problem in bipartite graphs. The switch state can be described as a bipartite graph $G = [V, E]$ (see Fig. 5) in which the graph vertices in set V are partitioned in two subsets: subset V_I, whose elements v_{Ik} correspond to input interfaces, and subset V_O, whose elements v_{Ok} correspond to output interfaces. Edges indicate the needs for cell transfers from an input to an output (an edge from v_{In} to v_{Om} indicates the need for cell transfers from input n to output m), and can be labeled with a metrics that will be denoted by w_{nm}. The adopted metrics is a key part of the scheduling algorithm; it can be equal to 1 to simply indicate that at least one cell to be transferred exist, or it could refer to the number of cells to be transferred, to the time waited by the oldest cell, or to other state indicators.

A matching M is a selection of an admissible subset of edges. A subset of edges is admissible if no vertex has two connected edges; this means that it never happens that two cells are extracted from the same input, or that two cells are transferred to the same output. A matching has *maximum size* if the number of edges is maximized; a matching has *maximum weight* if the sum of the edge metrics is maximized.

The different IQ cell switch architectures that we shall discuss differ in their scheduling algorithms, hence in their heuristic matching algorithms. The need for good matching algorithms derives from the fact that the optimal solutions of the problem have very high complexity. The complexity is $O(N^3)$ for the maximum weight matching algorithm, that can be proved to yield the maximum achievable throughput using as metrics either the number of cells to be transferred, or the time waited by the oldest cell; it is $O(N^{5/2})$ for the simpler and less efficient maximum size matching algorithm.

The $N \times N$ matrix whose elements are the edge metrics in graph $G = [V, E]$ is called the *weight matrix* $\mathbf{W} = [w_{ij}]$. $\mathbf{W}$ varies with time, according to the changes in the system parameters from which its elements are computed. When necessary, denoting by k the current slot, we shall write $\mathbf{W}(k) = [w_{ij}(k)]$. We assume $w_{jk} = 0$ for missing edges in G, i.e., when no cells from input j to output k are waiting in input queues.

3.2 Classification of Cell Scheduling Algorithms

A number of scheduling algorithms for IQ switch architectures have appeared in the technical literature. In this paper we consider four such proposals, namely iSLIP [3], iOCF [12], MUCS [2] (in its weighted version), and RPA [4]. iSLIP has been chosen for its simplicity, iOCF due to its metrics based on cell age, RPA for both simplicity and good performance under unbalanced traffic patterns, and MUCS for the very good performance obtained in different traffic scenarios.

The reader is referred to the original works for a detailed description of these algorithms. In this section we recall the general taxonomy for scheduling algorithms proposed in [6], and classify accordingly the considered proposals.

The output of the scheduling algorithm at slot k is a cell transfer matrix $\mathbf{T}(k) = [t_{ij}(k)]$, whose elements provide the result of the matching computation:

$$t_{ij} = \begin{cases} 1 \text{ if a cell is transferred from } i \text{ to } j \\ 0 \text{ otherwise} \end{cases} \tag{1}$$

Any IQ scheduling algorithm can be viewed as operating according to three phases:

1. *Metrics computation.* Computation of the weight matrix $\mathbf{W}(k) = [w_{ij}(k)]$. Each one of the possible N^2 edges in the bipartite graph is associated with a metrics depending on the state of the corresponding queue (the metrics associated with the edge from v_{Ii} to v_{Oj}, w_{ij}, depends on the content of Q_{ij} at slot k). This metrics will act as a priority for the cell transfer.

2. *Heuristic matching.* Computation of the cell transfer matrix $\mathbf{T}(k) = [t_{ij}(k)]$. This phase must try to maximize the following sum:
$$\sum_{i=1}^{N}\sum_{j=1}^{N} t_{ij}(k) w_{ij}(k) \tag{2}$$
with the constraints:
$$\sum_{i=1}^{N} t_{ij}(k) \leq 1 \qquad \sum_{j=1}^{N} t_{ij}(k) \leq 1 \tag{3}$$
Since the cost for the computation of the optimum matching (maximum size or maximum weight) is too high, all scheduling algorithms resort to heuristics with variable effectiveness and complexity. When the matching is not optimum, but no cells can be added without violating the constraints (3), the terms *maximal size matching* and *maximal weight matching* are used in the literature.
3. *Contention resolution.* In the execution of the heuristic algorithm for the determination of a maximal match, a strategy is necessary to solve contention due to edges with equal metrics, source or destination. The contention resolution typically is either random (RO, random order), or based on a deterministic scheme; round robin (RR) and sequential search (SS) are frequent choices in the latter case, the difference lying in the starting point chosen in the selection process, which is state-dependent for RR, and state-independent for SS.

The first phase is preliminary to the other two, that are instead interlaced.

As we shall see in Section 4.1, the phase that has the most profound impact on performance is the metrics computation, whereas the different heuristics to obtain good matches have a deep impact on the algorithm complexity.

As regards metrics, we consider the following alternatives:

– **QO** (*Queue occupancy*). In this case $w_{ij} = u(L_{ij})$ where $u(\cdot)$ is the unit step function. This is the metrics adopted by iSLIP. The adoption of this metrics leads to the search for a maximal size match. All other metrics lead to the search for maximal weight matches, which can differ in the weights.
– **QL** (*Queue length*). The metrics in this case is the number of cells in the queue: $w_{ij} = L_{ij}$. This is the metrics adopted by RPA.
– **CA** (*Cell age*). The metrics in this case is the time already spent in the queue by the cell at the queue head. This is the metrics adopted by iOCF.
– **ML** (*MUCS Length*). MUCS uses a metrics that is derived from queue lengths as:
$$w_{ij} = \frac{L_{ij}}{\sum_{k=1}^{N} L_{ik}} + \frac{L_{ij}}{\sum_{k=1}^{N} L_{kj}} \tag{4}$$

As regards the heuristic matching, the choices adopted in the considered IQ scheduling algorithms are the following:

Table 1. Characterization of the considered IQ scheduling algorithms

Algorithm	Metrics	Heuristics	Contention resolution	
			Input	Output
iSLIP	QO	IS	RR	RR
iOCF	CA	IS	RO	RO
MUCS	ML	MG	SS	RO
RPA	QL	RV	RO	SS

- **IS** (*Iterative search*). In this case, during a first step, each input interface sends all its transmission requests with the associated metrics to the relevant output interfaces ($w_{ij}(k)$ is sent from input interface i to output interface j). These select one among the arriving requests by choosing the largest metrics value, and resolving ties according to a contention resolution scheme (output contention). The accepted requests are then sent back to the input interfaces. If an input interface receives more than one acceptance, it selects one by choosing that with the largest metrics value, and resolving ties according to a contention resolution scheme (input contention). The successive steps are equal to the first one, but they concern only the transmission requests from input interfaces that received no acceptance, as well as those that were satisfied in previous steps (the repetition of these requests is necessary to progressively freeze the match configuration). This heuristic is adopted by iSLIP and iOCF.
- **MG** (*Matrix greedy*). Consider the $N \times N$ matrix $\mathbf{W} = [w_{ij}]$. In this case the algorithm consists of (up to) N steps, in each of which the largest element(s) w_{ij} of $\mathbf{W}$ is (are) selected, and the corresponding cell transmissions are enabled (provided that no conflict arises if ties for the largest metrics exist; otherwise a conflict resolution is necessary) before reducing the matrix by eliminating the entries corresponding to enabled cell transfers. This is the heuristics adopted by MUCS.
- **RV** (*Reservation vector*). In this case the algorithm is based on a sequential access to a reservation vector with N records, where input interfaces sequentially declare their cell transfer needs and the associated metrics, possibly overwriting requests with lower metrics values. A second sequential pass over the vector allows the confirmation of requests, or the reservation of other transfers for the inputs whose original reservations were overwritten. This is the heuristics adopted by RPA.

Table 1 gives a synoptic view of the considered IQ scheduling algorithms.

3.3 Packet-Mode Scheduling Algorithms

The additional constraint in this case is to keep the cells belonging to the same packet contiguous also in output queues. To achieve this, the scheduling algorithm must enforce that, once the transfer through the switching fabric of the first cell of a packet has started towards the corresponding output port, no cells belonging to other packets can be transferred to that output, i.e., when an input is enabled to transmit the first cell of a packet comprising k cells, the input/output connection must be enabled also for the following $k-1$ slots.

4 Simulation Results

To evaluate the performance of IQ, OQ, and CIOQ switching architectures, we conducted quite a large number of simulation experiments. We report results only for packet switches with $N = 16$ input/output interfaces, assuming that all input/output line rates are equal, and that only uniformly distributed unicast traffic flows are present.

Note that in IQ switches, each input queue Q_{ij} can store a finite number of cells; when a cell directed to output j arrives at input i, and queue Q_{ij} is full, the cell is lost. No buffer sharing among queues is allowed.

We do not explicitly describe the arrival of IP datagrams at the packet-switch inputs. We instead model the arrival of cell bursts at the inputs of the internal cell-switch. These cell bursts are assumed to originate from the segmentation of a packet.

The cell arrival processes at input i, $A_i(k)$, is characterized by a two-state model. In the ON state a packet is being received. The duration in slots of the ON state, i.e., the size in cells of the packet, is a discrete random variable uniformly distributed between 1 and 192, the latter value coming from the Maximum Transmission Unit (MTU) of IP over ATM. In the OFF state, no cells are received. The probability that the OFF state lasts j slots is $P(j) = p(1-p)^j \qquad j \geq 0$ The average idle period duration in slots is $\mathrm{E_{OFF}} = (1-p)/p$. The parameter p is set so as to achieve the desired input load.

The performance indexes we consider are the cell delay, i.e., the time spent by cells in the cell-switch queues, and the packet delay, i.e., the overall delay of a packet, considering the ISM module, the internal cell-switch queues, the ORM module, and the final packet FIFO.

Since the aim of the performance evaluation is a comparison of IQ and CIOQ architectures with OQ switches, we usually consider *relative* performance indices, i.e., we divide the absolute value taken by a performance index in the case of IQ and CIOQ architectures by the value taken by the same performance index in an OQ packet switch loaded with the same traffic pattern.

4.1 Performance of IQ Switches

Cell-Mode Scheduling Fig. 6 (left plot) shows, for the four considered scheduling algorithms operating in *cell-mode,* curves of the normalized average packet delay. Note that normalized delay values are always larger than 1, i.e., IQ switches always yield longer delays, but differences are limited within a factor 2.5, for load smaller than 0.95. No losses were experienced with input queue lengths equal to 30,000 cells.

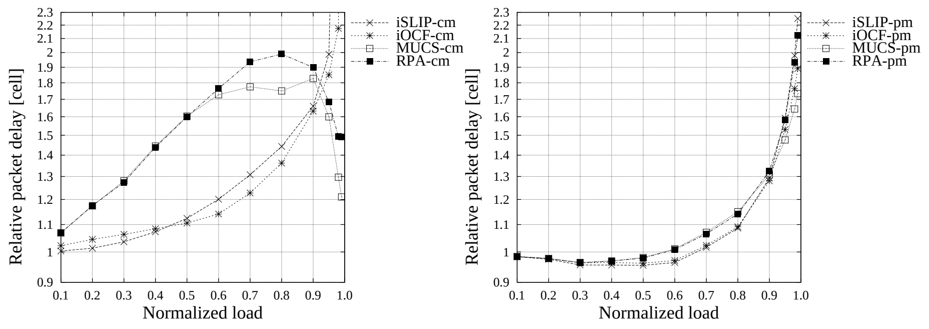


Fig. 6. Relative average packet delay for cell-mode scheduling (left) and packet-mode scheduling (right) in the uniform traffic scenario

For loads less than 0.9, MUCS and RPA generate longer delays than iSLIP and iOCF. This is mainly due to the QL and ML metrics, which tend to equalize queue lengths: when a large packet arrives at a particular queue, the queues (at other inputs) that store small packets directed to the same output suffer a temporary starvation. iOCF, instead, provides the lowest delays among weight-aware algorithms.

Packet-Mode Scheduling Performance results are significantly different if the *packet-mode* scheduling proposed in this paper is considered. Curves of the relative average delays are shown in Fig. 6 (right plot). Differences between the four algorithms are limited, and we observe (small) gains over OQ switches for loads up to around 0.6. IQ exhibits the largest delay advantage over OQ at loads around 0.4. To be fair in the comparison with OQ switches, we take into account, for IQ switches, delays due to ORM modules and to packet FIFOs, although these modules are not necessary in packet-mode operation, and at most one reassembly machine is active at each output, as discussed in Section 2.2. Cell delays in this same scenario, not shown here, are larger for all IQ architectures with respect to OQ at all loads.

The reductions in packet delays are interesting, specially if we consider the negligible additional cost of implementing packet-mode schedulers. This performance improvement, which is not very intuitive, can be shown to be related with the packet length distribution. Note that more complex scheduling algorithms, better tailored to the statistics of packet traffic, could probably provide even larger gains.

If the IQ switch architecture is further simplified to take advantage of packet-mode scheduling by removing ORM modules and output packet FIFOs, the gain in average packet delay over OQ becomes much larger, since the delay necessary to reassemble packets is avoided.

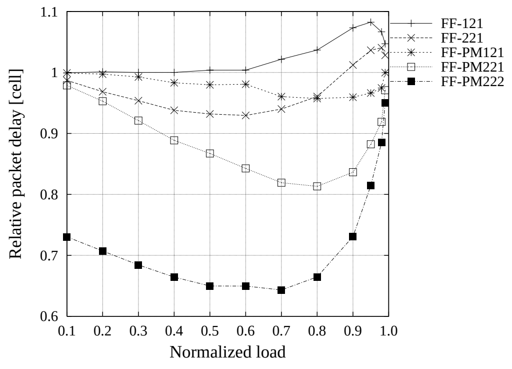


Fig. 7. Relative average packet delay in CIOQ architectures for the uniform traffic scenario

4.2 Performance of CIOQ Switches

This section presents simulation results for CIOQ switches using the FIFO–2 scheduling algorithm described in Section 2.1. A speed-up $S = 2$, is supposed to be available in the internal cell-switch. The capacity of all queues is taken to be unlimited for the results presented in this section, to be able to observe the unbalancement of queue lengths between input and output queues.

Fig. 7 plots curves of the average packet delay normalized to OQ values. The curves in the figure refer to the following switch setups, which differ in the values taken by the SPEEDUP-IP-IN and SPEEDUP-IP-OUT parameters (see Fig. 2 in Section 2.2), and in the operation mode (cell mode vs. packet mode) of the FIFO–2 scheduling algorithm (the extension of FIFO–2 to packet-mode operation is straightforward).

- **FF-121**: basic FIFO–2, SPEEDUP-IP-IN=SPEEDUP-IP-OUT= 1. The cells belonging to IP packets are fed to the cell-switch at the same speed of input and output lines.
- **FF-221**: basic FIFO–2, SPEEDUP-IP-IN= 2, SPEEDUP-IP-OUT= 1. The cells belonging to IP packets are fed to the cell-switch at twice the speed of input and output lines.
- **FF-PM121**: packet-mode FIFO–2, SPEEDUP-IP-IN=SPEEDUP-IP-OUT= 1. The cells belonging to the same packet are transferred to their output in contiguous slots. Note that up to two packets (received at different inputs) can be interleaved in output cell queues in this setup. An ORM module is therefore required at each output.
- **FF-PM221**: packet-mode FIFO–2, SPEEDUP-IP-IN= 2, SPEEDUP-IP-OUT= 1. The packet-mode FIFO–2 scheduling algorithm has in this case the additional constraint that packets cannot be interleaved in output cell queues: this becomes possible without performance degradation because of SPEEDUP-IP-IN= 2, since two cells of the same packet can be switched to their output

in one slot time. ORM modules are therefore not strictly necessary in this case: they are kept for a fair comparison with the other setups.
- **FF-PM222**: packet-mode FIFO–2, SPEEDUP-IP-IN= 2, SPEEDUP-IP-OUT= 2. The same constraints of FF-PM221 apply to the FIFO–2 algorithm: packets are not interleaved in output queues.

In the notation above, "FF" stands for FIFO, "PM" stands for packet-mode (recall the different constraints on the scheduling algorithm when SPEEDUP-IP-IN equals 1 or 2), and the three digits refer to the values taken by SPEEDUP-IP-IN, S, and SPEEDUP-IP-OUT, respectively.

As expected, average packet delays at low loads (not plotted here) were observed to be around twice the average packet duration for the architectures with SPEEDUP-IP-OUT= 1 (where one ISM and one ORM module must be traversed by each packet), and to be around 1.5 times the average packet duration for FF-PM222 (for which the ORM can be traversed by two packets in one packet time, i.e., at double speed).

The curves in Fig. 7 show that all the considered architectures operating in packet-mode have better performance than OQ packet switches. As previously observed, IQ and CIOQ switches provide extra advantages over OQ switches when variable-size packets are considered.

Note that the FIFO–2 algorithm in packet-mode can be more efficient than output queueing also when no speed-up exists between the IP line interfaces and the internal cell-switch (i.e., for FF-PM121). The best-performing setups are FF-PM221 and FF-PM222 which avoid output packet interleaving. Delay ratios can be as small as 0.64 (i.e., 1.5 times in favor of CIOQ) when the ORM modules are present, and lower values can be achieved for the optimized architecture.

FF-121 exhibits delays slightly larger that OQ. Note that FF-121 can be considered as a simplified version of OQ, in which at most 2 (instead of N) cells can reach an output in a slot time.

It is also interesting to observe that FF-221 behaves better that FF-121, mainly thanks to the fact that the cells belonging to the same packet are kept closer to each other in the transfer through the switch.

5 Conclusions

The paper focused on architectures and scheduling algorithms for IQ and CIOQ packet switches, comparing them with OQ switches in the case of variable-size data units.

Similar performance results were observed for IQ, CIOQ, and OQ, with a tradeoff between the complexity of the scheduling algorithm on one side, and hardware complexity due to the internal speed-up on the other side.

Packet-mode scheduling in IQ and CIOQ architectures makes OQ architectures scarcely attractive for the implementation of future-generation IP routers.

CIOQ architectures with small speed-up factors are particularly promising. They provide better performance than output queued architectures at lower hardware costs.

References

1. Anderson T., Owicki S., Saxe J., Thacker C., "High speed switch scheduling for local area networks", *ACM Transactions on Computer Systems*, vol. 11, n. 4, Nov. 1993, pp. 319-352
2. Duan H., Lockwood J.W., Kang S.M., Will J.D., "A high performance OC12/OC48 queue design prototype for input buffered ATM switches", *IEEE INFOCOM'97*, Kobe, Japan, April 1997
3. McKeown N., "iSLIP: a scheduling algorithm for input-queued switches", *IEEE Transactions on Networking*, vol. 7, n. 2, Apr. 1999, pp. 188-201
4. Ajmone Marsan M., Bianco A., Leonardi E., Milia L., "RPA: A flexible scheduling algorithm for input buffered switches", *IEEE Transactions on Communications*, vol. 47, n. 12, Dec. 1999, pp. 1921-33
5. McKeown N., Mekkittikul A., "A practical scheduling algorithm to achieve 100% throughput in input-queued switches", *IEEE INFOCOM'98*, New York, NY, March 1998
6. Ajmone Marsan M., Bianco A., Filippi E., Giaccone P., Leonardi E., Neri F., "On the behavior of input queuing switch architectures", *European Transactions on Telecommunications*, vol. 10, n. 2, Mar. 1999, pp. 111-124
7. Chuang S., Goel A., McKeown N., Prabhakar B., "Matching output queueing with a combined input/output-queued switch", *IEEE Journal on Selected Areas in Communications*, vol. 17, n. 6, June 1999, pp. 1030-39
8. Charny A., Krishna P., Patel N.S., Simcoe R., "Algorithms for providing bandwidth and delay guarantees in input-buffered crossbars with speedup", *6th International Workshop on Quality of Service (IWQoS'98)*, Napa, CA, May 1998
9. Ajmone Marsan M., Leonardi E., Mellia M., Neri F., "On the stability of input-buffer cell switches with speed-up", *IEEE INFOCOM 2000*, Tel Aviv, Israel, March 2000
10. Dai J.G., Prabhakar B., "The throughput of data switches with and without speedup", *IEEE INFOCOM 2000*, Tel Aviv, Israel, March 2000
11. Karol M., Hluchyj M., Morgan S., "Input versus output queuing on a space division switch", *IEEE Transactions on Communications*, vol. 35, n. 12, Dec. 1987, pp. 1347-1356
12. Mckeown N., Mekkittikul A., "A starvation free algorithm for achieving 100% throughput in an input queued switch ", *ICCCN'96*, Washington, DC, October 1996

Packet Discard Schemes for Differentiated Services Networks with ATM Switching Systems*

Maurizio Casoni

Dept. of Engineering Sciences - University of Modena and Reggio Emilia
Via Vignolese, 905 - 41100 Modena - Italy
casoni@dsi.unimo.it

Abstract. In this paper two packet discard schemes are proposed with the goal to provide incoming data flows different levels of quality of service. In particular, the assured forwarding in the Differentiated Services per-hop-behavior, as reported in the Request For Comments 2597, is considered as reference scenario.
The different levels of quality of service imply different portions of bandwidth assigned to data flows which share that same output link. Good performance in terms of throughput and fairness in bandwidth sharing among equal priority flows are also considered as the requirements to meet.
The proposed schemes are also evaluated when end-to-end transport protocols, such as TCP, are employed so as to give some informations about their impact on applications.
These dropping schemes are shown to provide good results and to represent a relatively low complex solution for dealing with the most demanding data flows during congestion intervals. Numerical results are reported and discussed for homogeneous and non homogeneous data sources.

1 Introduction

TCP/IP networks do not guarantee any type of quality of service (QoS) since their delivery service, “best effort”, implies that each IP data flow can have an indeterminate level of packet loss, delay, out-of-sequence. Since many multimedia applications have quite stringent requirements in terms of delay, loss and miminum bandwidth, a certain number of proposals have come out to provide some QoS to applications running over TCP/IP [1]. According to [2] a possible classification of models for service differentiation is the following: relative priority marking, service marking, label switching [3], integrated services [4]/RSVP [5] and static per-hop classification.

On the other hand, Asynchronous Transfer Mode (ATM) [6], chosen by ITU-T as the basic transfer mode for the Broadband Integrated Services Digital

* This work has been carried on by the author at D.E.I.S. - University of Bologna and has been supported by M.U.R.S.T. within the framework of the national project “Techniques for the QoS guarantee in multiservice IP networks”.

Network (B-ISDN), networks are supposed to carry many different types of traffic with different bandwidth requirements. Applications range from continuos ones transmitting informations at fixed rates to highly bursty which send large blocks of data at high peak rates but with long-term average rates relatively low and the different and changing characteristics of service requirements of current and emerging Internet applications push for a flexible network architecure.

The unpredictability of the human user and the high peak transmission rate required to provide acceptable interactive response imply that one must accept either a low network utilization or the possibility of overload periods during which the traffic sent to some network links exceeds their capacity. The fact that end-to-end protocols, such as TCP/IP, fragment their data in packets containing many ATM cells makes the impact of overload periods very dangerous, since a single lost cell can lead to the loss and retransmission of an entire transport level packet. Available bit rate (ABR) services are controlled by rate-based mechanisms but they work on round trip time intervals and so some other mechanisms must be implemented to cope with congestion occurring in short time intervals [7]. Packet discard schemes [8] [9] are ATM buffer management techniques to ensure high end-to-end throughputs [10] and fairness [11] [12] for bursty data applications during overload periods. Packet discard schemes are implemented in output buffered ATM switches, either output queued or combined input/output queued [13], and perform their task by selectively discarding cells from the incoming information flows.

An important issue is then to study how a router must treat packets belonging to different classes of service. The main approaches so far proposed can be summarized as packet or threshold dropping and priority scheduling. With the former, routers have one FIFO shared buffer among packets of all classes per output link while with the latter routers have separate logical queues for each class and some algorithm drives the scheduler for packet forwarding. Thus, whereas threshold dropping realizes service differentiation by means of thresholds, priority scheduling does it by properly serving packets of separate queues.

For instance, several approaches to congestion control and fairness in ATM switches are based on the implementation of per-VC queueing and several algorithms and designs for the output controller have been proposed [14] [15]. On the other hand, some recent works have underlined possible weaknesses of traditional packet discard techniques, such as random early detection [16], when employed in real networks [17]. Other works, such as [18], confirm the effectiveness of early packet discard to improve the throughput or, generally, the benefits in terms of complexity when buffer management schemes based on thresholds are employed with respect to scheduling mechanisms [19]. In conclusion, more efforts must be made to better understand the performance of these techniques and, in particular, to develop ever improved solutions.

In this paper the reference architecture is the one described in [2] ed in particular the focus is on the study of an effective mechanism to provide different levels of drop precedence in Differentiated Services (DS) nodes when ATM switching systems are employed. In a DS network, nodes can be divided into two classes depending on their position, at the edge or in the core of the network. Some authors have proposed and evaluated different priority schedulers and threshold dropping schemes for edge and core routers [20].

The Assured Forwarding (AF) Per-Hop-Behavior (PHB) [21] aims at providing delivery of IP packets in four independently forwarded classes. Each AF class is in each DS node allocated some portion of the forwarding resources, i.e. buffer space and bandwidth. Within each AF class IP packets are marked with one of three possible drop precedence values. If congestion occurs in a DS node (which may be equipped with an ATM switching architecture), this precedence determines how a packet has to be treated within the AF class. A congested DS node then must try to discard packets in accordance to these precedence values.

In this paper two packet discard schemes are proposed to properly manage in congested DS nodes the flows of packets depending on their drop precedence level within their AF class. One of these schemes can be employed in enterprise networks where two drop precedence levels are enough while the other can be used in DS domains where three drop levels are required. Their goal is also to mantain high throughput and fair exploitation of the bandwidth of the output link among equal-priority flows. The throughput is studied both on link-by-link basis and on end-to-end when TCP is assumed as transport level protocol. In fact, in wide area data networks like Internet congestion control mechanisms have a fundamental role for the global functioning [22]. The performance of packet discarding schemes in ATM networks have been widely investigated [23] [10] while their effects on the end applications have not definitely been shown yet, even if some works have been presented [24] [25]. Thus, it becomes important to determine the influence of buffer management techniques on the end-to-end performance in presence of end-to-end congestion control.

The reported results, obtained from the simulation of a single switch output controller, show that the main goals are met so that these schemes can be considered possible solutions for implementing service differentiation in congested DS nodes.

The paper is organized as follows: section 2 shows the reference system model, section 3 describes the first proposed discarding scheme suitable for enterprise networks and in section 4 the second scheme to be implemented in DS node belonging to large domains is explained. Section 5 reports some numerical results about the two proposed discarding schemes for different scenarios and traffic patterns; finally section 6 closes the paper with a drawing of the achieved outcomes.

2 System Model

The reference architecture is the one described in [2] and the goal of this paper is to study a possible candidate as packet dropping scheme for providing different levels of drop precedence in DS nodes. The assured forwarding PHB aims at providing delivery of IP packets in four independently forwared AF classes. Each AF class is in each DS node allocated some portion of the forwarding resources, i.e. buffer space and bandwidth. Within each AF class IP packets are marked with one of three possible drop precedence values. If congestion occurs in a DS node (equipped with an ATM switching architecture), this precedence determines how a packet has to be treated within the AF class. A congested DS node then must try to discard packets in accordance to these precedence values.

It is worthwhile to underline that packets in a AF class must be forwarded independently from packets of other AF classes, meaning that aggregation of packets belonging to different classes is not allowed. Also, a DS node must be able to deal with three drop precedence levels but in some cases they must yield at least two levels, e.g. in enterprise networks where congestion is rare, whereas three levels are desirable in DS domains.

The implementation of the AF class must avoid long-term congestion within each class, while it is allowed short-term congestion resulting from bursts. This implies the employment of an active queue management algorithm for providing the three drop precedence levels within each AF class. This algorithm can be thought to work in cooperation with a priority scheduling algorithm which has the task to manage the four AF classes. First proposals and evaluations begin to be published [20].

The three different drop precedence levels within each AF class are identified by different codepoints. The drop precedence levels are defined as Low Drop Precedence (LDP), Medium Drop Precedence (MDP) and High Drop Precedence (HDP), listed in decreasing order of importance, i.e increasing discarding "probability".

In the following two packet discard schemes are proposed to properly manage in congested DS nodes the flows of packets depending on their drop precedence level within their AF class. One of these schemes can be employed in enterprise networks where two drop precedence levels are enough while the other can be used in DS domains so as to provide three drop levels. Their goal is also to mantain high throughput both at link-by-link and at end-to-end level and to provide a fair exploitation of the bandwidth of the output link among equal-priority flows.

3 Packet Discard for Enterprise Networks

Some packet discard schemes have already been presented [8] [9] and evaluated [10] [11] [12] but none of them can discriminate among different classes of service. Here we assume to divide incoming flows, associated to VCs, between low and high priority (two classes of service) which can be respectively mapped in HDP and LDP levels of a DS node.

The proposed scheme employs two thresholds, b_l (low) and b_h (high), and let B denote the number of cells the buffer can contain. The buffer controller tries to keep the occupancy level between b_l and b_h by activating or deactivating the VCs at packet boundaries according to their priority level. The goal of this packet discarding scheme is to provide higher priority VCs (i.e. cells, i.e packets) with most of the bandwidth, to optimize the bandwidth exploitation and to reach at the same time a good fairness in bandwidth sharing among equal priority VCs. VC de/activation depends on the number of successfully transmitted cells during a time interval, called the current window, which is compared with the number transmitted by the other VCs. Since the buffer occupancy level has a cyclic behaviour [10], two thresholds aim at improving the performance by increasing the number of packets successfully transmitted during each cycle. A virtual circuit is defined to be active if the buffer controller has decided to accept the cells of the current packet; otherwise, it is inactive. In addition, if any cell of the packet must be discarded due to queue overflow, the remainder of that packet is discarded (tail packet discard).

Let n_i be the number of transmitted cells by VC i in the current window; A_m (I_M) be the smallest (largest) n_i from the set of currently active (inactive) VCs. The scheme here proposed is similar to the one presented in [11] with some variants to take priorities into account. Keeping in mind that during congestion all VCs send packets continually one beside the other, the scheme works as follows:

1. if the current buffer level exceeds b_h, at the packet boundaries of VC J make J inactive and always discard HDP cells;
2. if the current buffer level is below b_l, at the packet boundaries of VC J make J active;
3. if the current buffer level is between b_l and b_h, at the packet boundaries of VC J the buffer level is falling, J is inactive, $n_j < A_m$ and J is high priority make J active;
4. if the current buffer level is between b_l and b_h, the level is rising, J is active, $n_j > I_M$ and J is low priority make J inactive;
5. in all other cases do not change the state of J.

First two rules try to avoid that buffer overflows and underflows for optimizing the throughput and to save buffer space for high priority data. Third and fourth ones try to obtain fairness for equal priority VCs by turning on and off the VCs that have sent fewer or more cells in the current window, with respect to A_m and I_M. To handle fairness, the proposed scheme must keep the information regarding bandwidth usage of each VC in the current window. Also, it must maintain a record of the evolution of the buffer occupancy level in time in order to be able to decide if it is rising or falling.

Considering a homogeneous situation in which r VCs transmit data continually at a normalized rate of λ (that is, λ is the fraction of the link bandwidth required by a single virtual circuit), $r\lambda$ is defined as the overbooking ratio and for an overloaded link $r\lambda > 1$. It is assumed that all packets of all virtual circuits contain ℓ cells and let $k = \lfloor 1/\lambda \rfloor$ be the maximum number of virtual circuits that the link can handle without loss. The values of B, b_l, b_h largely determine the goodput, defined as the fraction of the link's capacity used to carry complete packets, that it is possible to achieve on a link that has a buffer managed by this packet discard scheme. In the second half of the paper another definition of goodput is used in order to take into account not only the link layer throughput but also the end-to-end throughput when a transport layer protocol is employed.

If b_l and b_h are properly set to avoid overflow and underflow and the difference $(b_h - b_l)$ is large enough, it is reasonable to foresee a great difference in bandwidth usage between high and low priority data and a quite good degree of fairness among equal priority VCs. The reason is that in this condition the scheme can turn on and off the VCs depending on priorities and bandwidth usage without penalizing throughput.

4 Packet Discard for DS Domains

In addition to the parameters n_i, A_m and I_M previously introduced, here one more is defined: A_{m-LDP}. It is used in a similar way as A_m but it regards packets belonging to the Low Drop Precedence level only; the goal is to manage the fairness of the low drop precedence level data flows separately from the others.

The second scheme adds some variants to the previous in order to manage three levels of priorities. The scheme works as follows:

1. if the current buffer level exceeds b_h, at the packet boundaries of VC J make J inactive and always discard HDP cells;
2. if the current buffer level is below b_l, at the packet boundaries of VC J make J active;
3. if the current buffer level is between b_l and b_h, the level is rising, J is active, $n_j > I_M$ and J is either HDP or MDP make J inactive;
4. if the current buffer level is between b_l and b_h, at the packet boundaries of VC J the buffer level is falling, J is inactive, $n_j < A_m$ and J is MDP make J active;
5. if the current buffer level is between b_l and b_h, at the packet boundaries of VC J, J is inactive, $n_j < A_{m-LDP}$ and J is LDP make J active;
6. in all other cases do not change the state of J.

The main characteristics here are that LDP data flows are never made inactive when the buffer level is below the threshold b_h and that the fairness is managed separetely from MDP and LDP flows.

For this scheme performance will be shown also when a transport layer protocol is employed between the end applications so as to get some outcomes of its end-to-end influence. Transport layer protocols such as user datagram protocol (UDP) and transmission control protocol (TCP) allow hosts to distinguish between multiple applications by means of port numbers.

An evolution of this basic scheme is also here presented in order to take into account the window based functioning of TCP. The discarding policy now takes into account the number of packets already discarded within the same transmission window in order to decide whether to discard also the current packet or not. Adding this information to the discarding policy aims at improving the end-to-end TCP throughput which will be compared with the throughput on the output link.

The authors in [25] have determined some relations between two successive losses within the same TCP connection in order to avoid timeout. In particular, if only two segments are lost within a window, the window must be greater than or equal to 10 segments; when three losses occur, the first and the second one must be spaced far enough to allow the third one to be fast retransmitted while when TCP looses 4 segments from the same window a timeout is unavoidable. Therefore, the packet discard scheme previously detailed is now slightly changed to avoid, when possible, to discard a packet from a certain VC if this VC has had a "recent" packet loss, with the aim to reduce the probability of loosing more than three segments within the same window.

In particular, the rule number 3 of the basic scheme is changed as follows:

– if the current buffer level is between b_l and b_h, the level is rising, J is active, $n_j > I_M$, J is either HDP or MDP and the number of TCP segments already lost in the current window is < 2, make J inactive

The rationale of this addition consists of trying to delay as long as possible another loss in the current TCP window to decrease the probability of timeouts, whenever two or more losses have already occured. In section 5 this scheme will be called version 2.

5 Numerical Results

A simulation program of the algorithms previously described has been developed to study the performance in terms of per VC bandwidth usage, output link goodput, fairness in bandwidth usage and end-to-end, i.e. TCP, goodput.

TCP goodput is defined as the fraction of the link's capacity used to carry complete IP packets containing TCP segments correctly accepted by the TCP on the receiver host.

The bandwidth usage is the figure of merit for the level of quality assigned to each VC in presence of congestion and it must respect the drop precedence level of the related user data flow. The discarding schemes will then be tested on this parameter as well. Besides, it is worth reminding that the proposed schemes employ two thresholds which aim at increasing the number of succesfully transmitted packet per cycle, thus improving the goodput.

The simulation requires to set up several parameters: the simulation duration is 1000 packet times long, the current window is 200 packet times. All numerical results assume a TCP maximum segment size equal to 960 bytes, which gives IP datagram of 1000 bytes (20 bytes TCP header + 20 bytes IP header), maximum congestion window size is 64 Kbytes. Also, ℓ is set to 21 to represent one ATM adaptation layer (AAL) 5 packet.

Assuming a 600 Mbit/s output link, the bandwidth usage is considered in the overbooking ratio range up to 3 Gbit/s which is a likely scenario, which means $r\lambda$ in the range [1..5]. The schemes have been evaluated in different operating scenarios and traffic patterns, namely, with homogeneous sources sending data at the same rate, or normalized rate λ, and with non homogeneous sources which have been grouped in two different sets characterized by different data rates, λ_1 and λ_2 respectively.

Next paragraphs will report some numerical results for these two main scenarios in enterprise networks and DS domains.

5.1 Packet Discard for Enterprise Networks

In enterprise networks, where congestion is supposed to be rare and very short, two drop precedence levels may be enough so that the reported results refer to the scheme described in section 3.

Homogenous Sources Two situations are here analyzed. Both refer to an homogeneous situation meaning that all VCs have the same rate. Case 1 has $k = 12$, $B = 512$, $b_h = 256$, $b_l = 64$ and 1/3 of the VCs are LDP. Case 2 corresponds to $k = 10$, $B = 812$, $b_h = 512$, $b_l = 128$ and 1/4 of the VCs are LDP. In all cases the LDP level VCs have been randomly chosen.

Figure 1 refer to case 1 and show the bandwidth usage for each VC for an overbooking ratio equal to 2. In Figure 1 LDP (high priority) VCs belong to the number set $(3, 6, 9, 11, 15, 18, 20, 23)$: most of the bandwidth is assigned to higher priority VCs and within the two classes of services a significant level of fairness is obtained.

Figure 2 refer to case 2 and show the bandwidth usage for each VC for an overbooking ratio equal to 3. In Figure 2 LDP VCs belong to the number set $(2, 5, 8, 11, 15, 18, 22, 26)$ and the bandwidth is again assigned mostly to high priority VCs with a reasonable fairness.

Table 1. Goodput as a function of the overbooking for the two analyzed cases

overbooking ratio =	2	3	4	5
Case 1	0.9772	0.9825	0.979	0.9786
Case 2	0.9977	0.9993	0.9995	0.998

Table 1 shows the goodput values reported for the two considered cases and underlines the effectiveness of the proposed scheme in optimizing the output link.

It is worth noting that since the overall goodput for each case is so high, the previous figures about bandwidth usage also represent the distribution of the goodput among VCs.

The results obtained mainly match the proposed target of assigning the output link to LDP VCs first, keeping the goodput as high as possible and trying to provide an equal sharing of the link among equal priority VCs.

Non Homogeneous Sources In this section the packet discard scheme is studied in a different scenario. Let us assume that VCs have two different rates, λ_1 and λ_2 with $\lambda_1 > \lambda_2$. It is also assumed that VCs still transmit packets of the same length, $\ell = 21$. As a consequence, the packet transmission time is different for VCs with different rates and $\ell/\lambda_1 < \ell/\lambda_2$. Therefore, during the transmission time $T_2 = \ell/\lambda_2$ more than one packet boundaries per each VC can occur for faster VCs. This means that faster VCs ask the buffer controller for more accesses to the buffer than slower ones and this may lead to some unfairness between fast and slow VCs.

In the following cases, high priority is assigned to faster VCs, i.e. they have a LDP level, assuming that they are used for time critical applications or because users simply pay for that. Of course, other assumptions and mixes can be made but they will not be discussed here.

In figure 3 $\lambda_1 = 0.2$, $\lambda_2 = 0.1$, $r_1 = 5$ and $r_2 = 30$ and the overbooking ratio is then $r_1\lambda_1 + r_2\lambda_2 = 4$; the high priority VCs $(2, 13, 23, 28, 33)$ are 1/7 of the total. Again, as desired, they get most of the bandwidth and HDP (low priority) VCs can take small portions of it.

Basically, all the bandwidth is taken by the r_1 sources and a quite good level of fairness is achieved among them. On the other hand, the r_2 sources get almost nothing but, actually, this is just what we want: priorities have been set up to assign resources, i.e. bandwidth, to specific users, in particular during the short congestion intervals.

5.2 Packet Discard for DS Domains

In large DS domains it is likely that congestion occurs more often in DS nodes than in enterprise networks. Thus, a more accurate packet discard scheme must be employed to take into account more than two drop precedence levels.

The following results have been obtained by using the two discarding schemes, basic and version 2, detailed in section 4, which can treat in a different way data flows belonging to three different priority levels, namely LDP, MDP and HDP levels. As previously mentioned, the schemes will be evaluated also on end-to-end basis when TCP is employed. Therefore two kinds of goodput will be discussed: ATM goodput, which is the goodput at link level, and TCP goodput, which is the end-to-end goodput.

Homogenous Sources Two situations are here analyzed as well. Both refer to an homogeneous situation, again meaning that all sources have the same rate.

Case 1 has $k = 12$, $B = 512$, $b_h = 256$, $b_l = 64$.

Case 2 corresponds to $k = 10$, $B = 812$, $b_h = 512$, $b_l = 128$.

Table 2 shows the goodput values reported for the two considered cases as a function of the overbooking ratio. First of all, from this table we can see the remarkable difference in performance between link and end-to-end level. While the proposed scheme provide almost 100% goodput at link level, for the TCP end-to-end connection the goodput is lower and drops in the range of $30 - 40\%$ as the overbooking ratio increases beyond $2 - 3$. However, it is worthwhile noting that the version 2 of the packet discard scheme manages to keep high TCP goodputs for overbooking ratios equal to 1.5 and 2, improving then the basic scheme, but as soon as the load goes beyond these values, it performes as the basic scheme because of heavy congestion. This last point is very important because many previous works have evaluated the influence of packet discard schemes on TCP connections with typically overbooking ratio values in the range of $1.5 - 2$.

The main conclusion here is that the proposed packet discard schemes can provide good results also on end-to-end basis but their performance in terms of end-to-end goodput rapidly deteriorate as soon as the node gets heavily congested, as opposed to link level goodput which is always very high and in many cases increases by increasing the overbooking ratio [10].

Figure 4 refer to case 1 and show the bandwidth usage for each VC for an overbooking ratio equal to 2. In this figure LDP (high priority) VCs belong to the number set $(2, 5, 10, 13, 17, 22)$ and have been randomly chosen. Also, the set $(3, 6, 9, 11, 15, 18, 20, 23)$ corresponds to MDP VCs.

The main result is that the scheme version 2 assignes the output link bandwidth respecting the drop precedence levels: almost 50% of the bandwidth is given to LDP flows at the link level, and even more at the TCP level, and fairly shared among them, roughly 8% each; the eight MDP flows get approximately 6% each; finally, also HDP flows get some bandwidth, roughly 4% all together.

Table 2. ATM and TCP Goodput as a function of the overbooking for the two analyzed cases

overbooking ratio =	1.5	2	3	4	5
Case 1 - ATM Gput	0.9994	0.9989	0.9982	0.9979	0.9981
Case 1 - TCP Gput (basic)	0.5853	0.7209	0.3832	0.3377	0.337
Case 1 - TCP Gput (version 2)	0.9778	0.7385	0.3723	0.3347	0.336
Case 2 - ATM Gput	0.9995	0.9994	0.9991	0.999	0.999
Case 2 - TCP Gput (basic)	0.8236	0.771	0.5587	0.4261	0.3987
Case 2 - TCP Gput (version 2)	0.9958	0.8917	0.5661	0.3869	0.3925

Figure 5 refer to case 2 and show the bandwidth usage for each VC for an overbooking ratio equal to 5. In this figure LDP (high priority) VCs belong to the number set $(2, 5, 10, 12, 15, 22, 25, 30, 32, 35, 42, 45)$ and again have been randomly chosen. Also, the set (3, 6, 9, 13,16,19,20, 23, 26,29, 33, 36, 39, 40, 43,46, 49,50) corresponds to MDP VCs. Also in this case, the proposed scheme respects the drop precedence levels: 52.8% of the bandwidth is given to the twelve LDP flows and fairly shared among them, roughly 4.4% each; the sixteen MDP flows get approximately $2-3$% each. Also HDP flows get some bandwidth, roughly 0.04% each, with 0.8% the amount for the whole HDP level. Again, it is important to underline the remarkable level of fairness obtained within each level.

Non Homogeneous Sources As in subsection 5.1, VCs may have two different rates, λ_1 and λ_2 with $\lambda_1 > \lambda_2$, and transmit packets of the same length, $\ell = 21$.

Figure 6 refers to the case with $k_1 = 5$, $k_2 = 10$, $r_1\lambda_1 + r_2\lambda_2 = 4$, $B = 512$, $b_h = 256$, $b_l = 64$, $(3, 6, 9, 13, 17, 20, 26, 30, 32)$ are the LDP VCs, randomly chosen again, and it shows how the goodput is distributed among the VCs at ATM and TCP level. The $r_1 = 5$ fast sources are in the set $(2, 13, 23, 28, 33)$: one of them has LDP level (number 13), two have a MDP level (number 2 and 28) and two a HDP level (23 and 33). It is worthwhile noting that the bandwidth has been assigned according to the drop precedence levels also in this case meaning that the proposed scheme can effectively support the presence of different rates sources. Actually, with respect to the results with homogenous sources, here there is more fluctation in the amount of assigned bandwidth within each level but, still, the difference among them is clear. As a matter of fact, considering the LDP sources, five get 6.3% the bandwidth, three the 5.8% and one the 3.9%, and the sum is roughly 52.8%; as regards the twelve MDP sources, they get on average the 3.8%, globally the 45.6%; finally the HDP sources share the remaining 1.6%, which is roughly 0.1% each.

Thus, basically the output link bandwidth is assigned to LDP and MDP sources, even if a small portion is given also to HDP level ones, and between them most of it is taken by LDP sources that is exactly what we want. In addition, a quite good level of fairness is achieved among flows belonging to the same drop precedence level.

It is important to note that in this case, overbooking ratio equal to 4, the goodput at cell level is 99.94% , while when TCP is taken into account it drops to 32.9%, underlining again the remarkable difference between link and end-to-end perfomance for high load values. Therefore, the values of bandwidth usage reported in the figure show how these goodputs distribute over the VCs.

6 Conclusions

In this paper some selective packet discard schemes have been proposed for managing congestion in output buffered ATM switches taking into account the class of service, or drop precedence level, of the incoming data flows. They have been employed in network nodes in two different scenarios, namely enterprise networks and differentiated service domains. Their goal is to provide output bandwidth to data streams in accordance to their priority and two or three priority levels have been considered. Also, they have to mantain high goodputs and fair exploitation of the bandwidth of the output link among equal-priority flows. The goodput has been studied both on link-by-link basis and on end-to-end when TCP is assumed as transport level protocol.

The analyzed schemes try to keep the buffer occupancy level between two thresholds as long as possible by de/activating the VCs depending on the number of successfully transmitted packets in a time window and on the drop precedence level, in order to provide high goodput and good fairness in bandwidth sharing among equal priority VCs. In addition, one scheme has been changed to take into account the number of TCP segment losses occured within a TCP congestion window so as to reduce the occurency of timeouts.

For the operating scenarios, homogeneous and non homogeneous, here taken into account the proposed schemes have been shown to meet the mentioned target and, in particular, to be capable of managing successfully different priority data flows without excessively penalizing throughput or fairness.

In conclusion:

- Packet discard schemes based on thresholds are effective to differentiate incoming traffic into two or three classes of service, to provide a reasonable fairness among equal priority flows and to keep a very high link level goodput;
- These schemes can provide good results also on end-to-end basis but their performance in terms of end-to-end goodput rapidly deteriorate as soon as the node gets heavily congested
- An overbooking ratio equal to 2 can be considered a threshold value to get good values of end-to-end goodputs

The above results have been obtained through simulation so that it is not straightforward to foresee the exact behaviour of the scheme in a real network. As a matter of fact, realistic scenarios can be more complex than the ones here presented and analyzed. Combinations of different packet lengths with many different data rates and priorities are possible and this is current work.

The subject of packet discarding schemes is still open to discussion about their effectiveness, as recent literature has underlined, so that many more investigations and possibly test-beds are necessary to better clarify this issue. However, these first results show that is possible to set up a packet discard scheme whose task is to provide class-dependent bandwidth assignment with acceptable fairness and reasonably high goodputs.

References

1. Stevens, W.R.: TCP/IP Illustrated. Vol.1, Addison Wesley, 1996.
2. Blake, S., Blake, D., Carlson, M., Davies, E., Wang, Z., Weiss, W.: An Architecture for Differentiated Services. Request for Comments 2475 (December 1998)
3. Callon, R., Swallow, G., Feldman, N., Viswanathan, A., Doolan, P., Fredette, A.: A Framework for Multiprotocol Label Switching. Internet Draft, draft-ietf-mpls-framework-02.txt (August 1997)
4. Braden, B., Clark, D., Shenker, S.: Integrated Services in the Internet Architecture: an overview. Request for Comments 1633 (June 1994)
5. Braden, R., Zhang, L., Berson, S., Herzog, S., Jamin, S.: Resource ReSerVation Protocol (RSVP) - version 1 functional specification. Request for Comments 2205 (September 1997)
6. De Prycker, M.: Asynchronous Transfer Mode: a Solution for B-ISDN. Ellis-Horwood, II edition (1993)
7. Jain, R.: Congestion Control and Traffic Management in ATM Networks: Recent Advances and a Survey. Computer Networks and ISDN Systems, vol. 28, no. 13 (October 1996) 1723-1738

8. Floyd, S., Romanow, A.: Dynamics of TCP Traffic over ATM Networks. Computer Communication Review, vol. 24, no. 4, (1994)
9. Turner, J.S.: Maintaining high throughput during overload in ATM switches. Proc. of Infocom '96, San Francisco, CA (USA) (1996)
10. Casoni, M., Turner, J.S.: On the Performance of Early Packet Discard. IEEE Journal on Selected Areas in Communications, special issue on Advances in ATM Switching Systems for B-ISDN, vol. 15, no. 5 (June 1997) 892-902
11. Casoni, M.: Fair Packet Discard as Improvement of Early Packet Discard. Proc. of Globecom'98, Sydney (AUS) (November 1998)
12. Chan, S., Wong, E.W.M., Ko, K.T.: Fair Packet Discarding for Controlling ABR Traffic in ATM Networks. IEEE Transactions on Communications, vol. 45, no. 8 (August 1997) 913-916
13. Chuang, S., Goel, A., McKeown, N., Prabhakar, B.: Matching Output Queueing with a Combined Input/Output-Queued Switch. IEEE Journal on Selected Areas in Communications, vol.17, no.6 (June 1999) 1030-1039
14. Benmohamed, L., Wang, Y.T.: A Control-Theoretic ABR Explicit Rate Algorithm for ATM Switches with Per-VC Queueing. Proc. of Infocom'98, San Francisco, CA (USA) (1998)
15. Chen, Y., Turner, J.S.: Design of a Weighted Fair Queueing Cell Scheduler for ATM Networks. Proc. of Globecom'98, Sydney (AUS) (November 1998)
16. Floyd, S., Jacobson, V.: Random Early Detection Gateways for Congestion Avoidance. IEEE/ACM Transactions on Networking (1993) 397-413
17. May, M., Bolot, J., Diot, C., Lyles, B.: Reasons not to deploy RED. Proc. of 7th International Workshop on Quality of Service, London (U.K.) (1999)
18. Goyal, R., Jain, R., Goyal, M., Fahmy, S., Vandalore, B.: Traffic Management for TCP/IP over Satellite ATM Networks. IEEE Communications Magazine (March 1999) 56-61
19. Guérin, R., Kamat, S., Peris, V., Rajan, R.: Scalable QoS Provision Through Buffer Management. Proc. of SIGCOMM '98, Vancouver (Canada) (1998) 29-40
20. Sahu, S., Towsley, D., Kurose, J.: A Quantitative Study of Differentiated Services for the Internet. KICS/IEEE Journal of Communications and Networks, vol. 2, no. 2 (June 2000) 127-137
21. Heinanen, J., Baker, F., Weiss, W., Wroclawski, J.: Assured Forwarding PHB Group. Request for Comments 2597 (June 1999)
22. Floyd, S., Fall, K.: Promoting the Use of End-to-End Congestion Control in the Internet. IEEE/ACM Transactions on Networking, vol.7, no.4 (August 1999) 458-472
23. Kim, Y., Li, S.: Performance Analysis of Data Packet Discarding in ATM Networks. IEEE/ACM Transactions on Networking, vol.7, no.2 (April 1999) 216-227
24. Suter, B., Lakshman, T.V., Stiliadis, D., Choudhury, A.K.: Design Considerations for Supporting TCP with Per-flow Queueing. Proc. of Infocom '98, San Francisco, CA (USA) (1998)
25. Cohen, R., Hamo, Y.: Balanced Packet Discard for Improving TCP Performance in ATM Networks. Proc. of Infocom 2000, Tel Aviv (Israel) (2000)

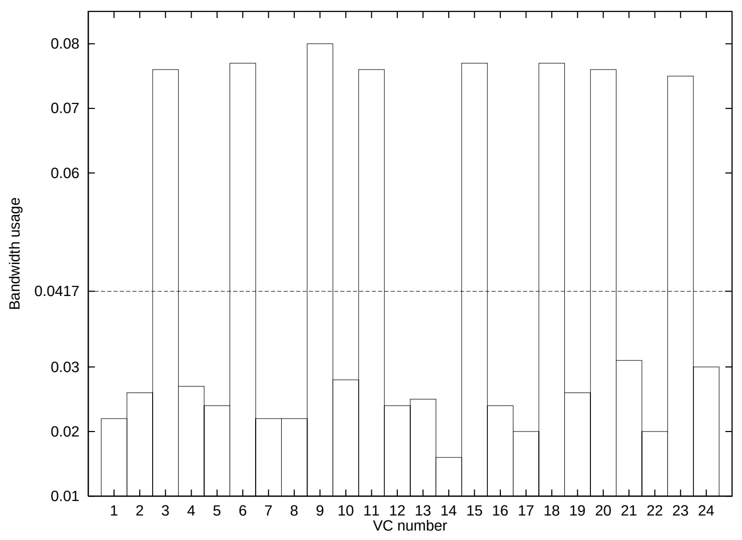


Fig. 1. Bandwidth usage for each virtual circuit; $k = 12$, $r\lambda = 2$, $B = 512$, $b_h = 256$, $b_l = 64$; dotted line represents the ideal sharing corresponding to 1/24 of the output link.

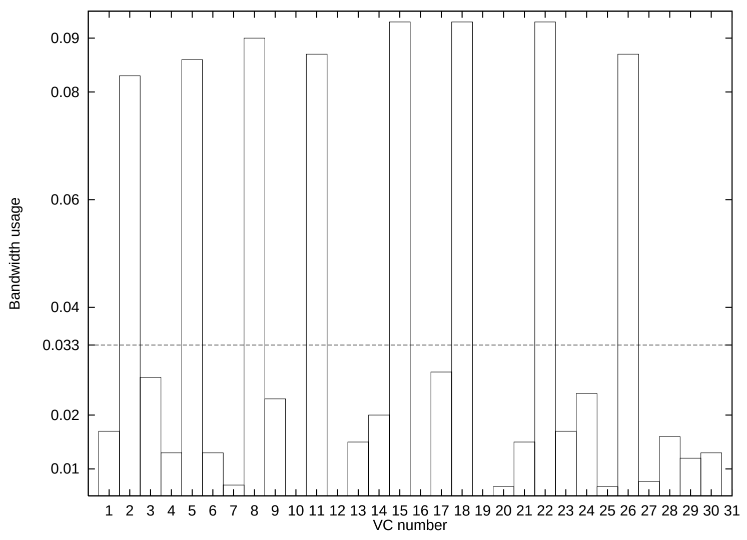


Fig. 2. Bandwidth usage for each virtual circuit; $k = 10$, $r\lambda = 3$, $B = 812$, $b_h = 512$, $b_l = 128$; dotted line represents the ideal sharing corresponding to 1/30 of the output link.

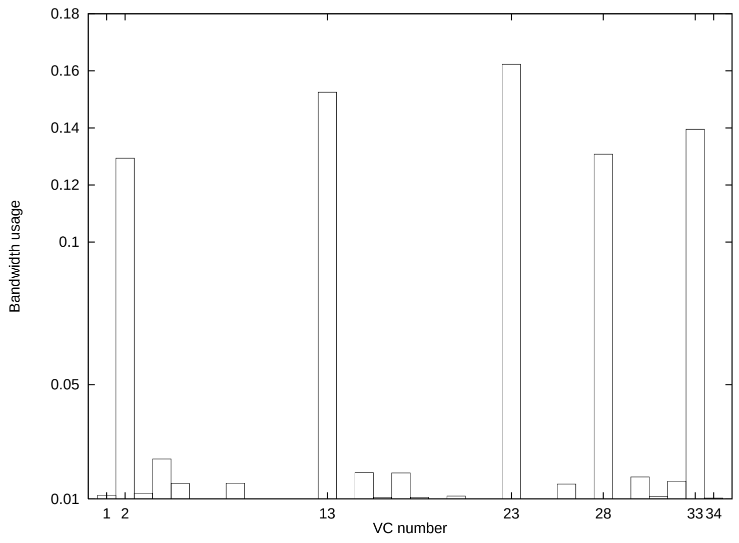


Fig. 3. Bandwidth usage for each virtual circuit; $k_1 = 5$, $k_2 = 10$, $r_1\lambda_1 + r_2\lambda_2 = 4$, $B = 512$, $b_h = 256$, $b_l = 64$; (2,13,23,28,33) are the high priority VCs.

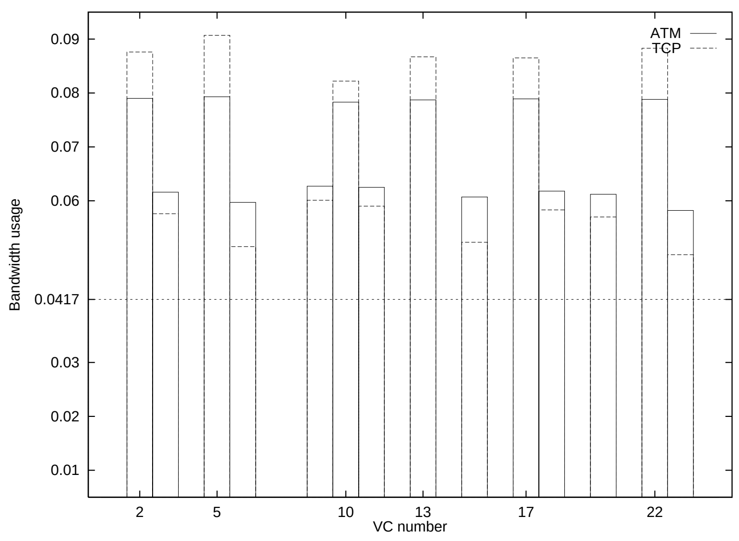


Fig. 4. Bandwidth usage for each virtual circuit at ATM and TCP level; $k = 12$, $r\lambda = 2$, $B = 512$, $b_h = 256$, $b_l = 64$; (2,5,10,13,17,22) are the low drop precedence VCs.

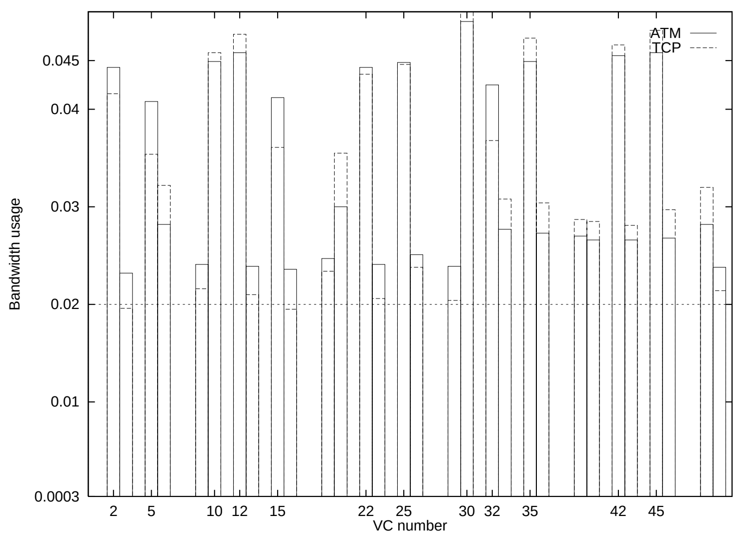


Fig. 5. Bandwidth usage for each virtual circuit at ATM and TCP level; $k = 10$, $r\lambda = 5$, $B = 812$, $b_h = 512$, $b_l = 128$; (2,5,10,12,15,22,25,30,32,35,42,45) are the low drop precedence VCs.

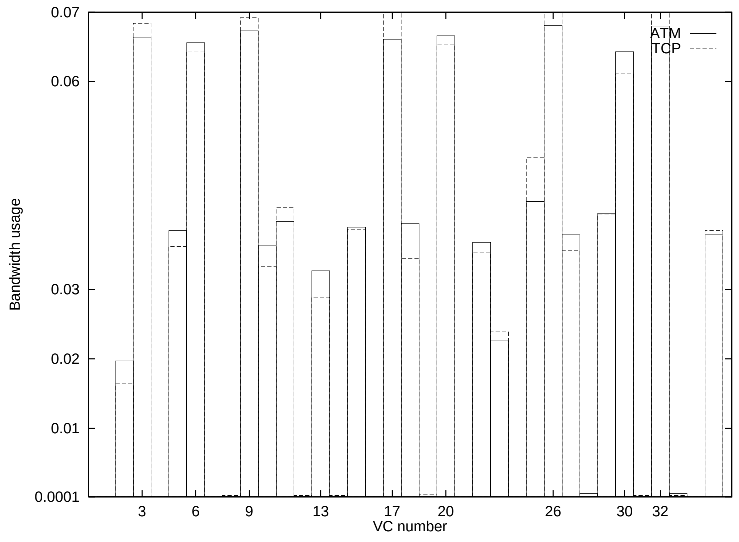


Fig. 6. Bandwidth usage for each virtual circuit at ATM and TCP level; $k_1 = 5$, $k_2 = 10$, $r_1\lambda_1 + r_2\lambda_2 = 4$, $B = 512$, $b_h = 256$, $b_l = 64$; (3,6,9,13,17,20,26,30,32) are the low drop precedence VCs.

Analysis and Simulation of WF^{2}Q+ Based Schedulers: Comparisons and Compliance with Theoretical Bounds

Nicola Ciulli[1] and Stefano Giordano[2]

[1] Consorzio Pisa Ricerche - META Center
corso Italia, 116 - 56125 Pisa, Italy
n.ciulli@cpr.it
http://www.meta.cpr.it
[2] Dept. of Information Engineering - University of Pisa
Via Diotisalvi 2 - 56126 Pisa, Italy
giordano@iet.unipi.it
http://wwwtlc.iet.unipi.it

Abstract. The statistical multiplexing of non-fixed-size packet flows with heterogeneous requirements onto a single network interface of a router gave rise to a number of different scheduling mechanisms. These algorithms attempts to work as close as possible to the ideal *"fluid model"*. One of the most effective proposals is the *Worst-case Fair Weighted Fair Queueing (WF^{2}Q)*; this pays its optimality with a great computational complexity and has been followed by more "operative" derivatives: the WF^{2}Q+ and S-SPFQ. The former has not been specified with a univocal algorithm, thus leaving space for a number of implementations: this article aims to analyse and compare the different algorithms coming out from the WF^{2}Q concept at both theoretical and "practical" level (by means of simulations). To this purpose, an on-purpose discrete-event simulator for packet schedulers has been implemented at the University of Pisa. The simulations also allowed to verify to what extent the operative versions of WF^{2}Q fulfill to the WF^{2}Q theoretical properties.

1 Introduction

Forwarding a number of packet flows (i.e. inter-related sequences of packets) through a single network interface in a router requires the use of proper statistical multiplexing disciplines (i.e. packet schedulers) to regulate the access of packets from multiple queues to the interface. The purpose is to provide different shares of the service resource (i.e. bandwidth) to the different flows and, for this reason, they can be considered the basic functions to implement a service differentiation over IP networks. Other queueing disciplines (such as discarding functions) may be used, and have been implemented in our IP-QoS trial, but for sake of conciseness the tests shown in this article focus only on the scheduling aspect.

A packet scheduling key problem rises from the basic fact that only one packet at a time can be transmitted through a network interface. This leads to consider as "merely ideal" the scheduler model where all the flows are served simultaneously (possibly at different "speeds" - i.e. shares of bandwidth). These models are often referred to as *Generalised Processor Sharing (GPS)* or *Fluid Fair Queueing (FFQ)* schedulers.

The problem of sharing link bandwidth among different flows (mapped into scheduler "connections") causes a number of alternative proposals aiming to realise the scheduling algorithm which better approximates the ideal behaviour of the GPS model.

1.1 The Problem of GPS Approximation

The "distance" of the approximation from its ideal model is measured through a set of properties and parameters, among which one of the most significant is the *"fairness"*, introduced later on. Here are listed some aspects under which a "real" packet scheduler may differ from the "ideal" one:

– since only one packet can be served each time, in different periods the whole link bandwidth is given to a single connection, and the link capacity is shared among the connections only in terms of their average values computed over "long" periods of time. (i.e. "long enough" to smooth such granularity in bandwidth allocation).
– In the GPS model, a packet entering a queue will wait for service for the time needed by the scheduler to *clear* the queue. In the real scheduler, the packet may experience a longer delay; e.g. if the packet enters an empty queue and the scheduler is busy, its transmission will not start immediately (as in the GPS model), but when the packet under service is finished.
– On the other hand, a packet may finish its service in the real scheduler sooner than in the GPS; e.g. if a packet enters an empty queue when the server is *idle*, its service time is L/C in the real scheduler (where L is the packet's size and C the link capacity), and L/r in the GPS (where r is the connection's service rate).
– As a consequence of the last two points, the connection queue occupancy in the real scheduler is different from the one in the GPS. The discrepancy between two corresponding queues should be limited and, possibly, small (the same applies to queueing delays).

1.2 Packet Scheduler "Fairness"

A *fair* scheduler manages the link bandwidth in such a way to give each connection an amount of service (i.e. time spent serving) proportional to its reserved rate, on *any* time interval. An *unfair* scheduler may offer a class, in some short periods, a service rate different from the allocated one. In real schedulers the shortest interval considered is the packet time, since during a packet service the whole link is owned by a single connection. The ideal behaviour is reached when

$w_i(t_1, t_2) = w_k(t_1, t_2)$ $\forall i, k \in B(t_1, t_2)$ (being $B(t_1, t_2)$ the set of backlogged connections and $w_i(t_1, t_2)$ is the service offered to class i, normalised to its allocated rate, in the interval $]t_1, t_2[$). The condition above is satisfied for any t_1, t_2 only by the GPS, whereas in a real system (since the granularity is the packet) it is never true at all and some indexes have been proposed to measure the scheduler behaviour in terms of fairness.

A good fairness index is the distance among the normalised services of the various connections on different time intervals. Another index is the *Worst-case Fair Index (WFI)* [1], which we focussed on; being r_i the service rate for class i, the WFI for scheduler S is defined as the value $C_{i,S}$ such that, for each packet, the following condition is true:

$$d_i^k \le a_i^k + Q_{i,S}(a_i^k)/r_i + C_{i,S} \ , \tag{1}$$

where a_i^k is the arrival time of the packet at the i^{th} connection's queue, d_i^k is its departure time and $Q_{i,S}(t)$ is the queue occupancy at time t^+ (thus, $Q_{i,S}(a_i^k)$ includes the newly arrived packet).

1.3 The WF^{2}Q Algorithm

The WF^{2}Q has been introduced as a fairer variation of the basic WFQ structure. The WFQ [2][3] belongs to the class of *sorted-priority* schedulers [1] and is based on the emulation of the *corresponding* [2] GPS system, by means of a *"System Potential"* (or *"Virtual Time"*) function (which measures the overall work carried out by the GPS). When the WFQ is ready for service it chooses, among all the head-of-line packets, the one finishing its service first in the GPS, if no other packets were to arrive after the start of the service (this is a *SFF policy: Smallest virtual Finish time First*). The WFQ allows a good packet interleaving, but it may start serving also packets which would not have gone under service in the GPS yet, resulting in "bursts" of service on a single connection (for some reservation layouts and traffic inputs the WFQ's WFI increases linearly with the number of connections).

In order to overcome this weak point, the WF^{2}Q's set of service eligible packets is reduced to those packets which would have already started their service in the GPS (in the WFQ it is the whole group of head-of-line packets). This is an *SEFF (Smallest Eligible virtual Finish time First)* policy, and allows a very fine-grain interleaving of packets from the different connections and, consequently, a greater fairness.

[1] The *sorted-priority* schedulers carry out a packet-by-packet scheduling according to some time-varying priority assigned to each connection and provides a better fairness with respect to the *frame-based* schedulers (which divide the time into frames, assign a time-slot to each connection and serve bursts of packets in it – a famous example of them is the *Weighted Round Robin (WRR)*).

[2] Two packet schedulers are *corresponding systems* if they have the same configuration (same set of connections - in terms of queue length, service rate), and same incoming traffic patterns.

In the remainder of this article, section 2 introduces some details on the WF^{2}Q-based scheduling algorithms and outlines the symbolism which will be used throughout the article. In section 3 some comparisons among packet schedulers are carried out.

2 Overview of WF^{2}Q-Based Scheduling Algorithms

2.1 Some History

As introduced in section 1.3, the main computational burden in the WFQ and WF^{2}Q implementations is the calculation of a system potential function. Thus, the WF^{2}Q proposal has been followed by many "operative" algorithms which try to approximate it. Among these schedulers, generically named as "WF^{2}Q+", some descend directly from the original WF^{2}Q+ definition [4], whereas one (the *Shaped Starting Potential Fair Queueing – S-SPFQ*) is the outcome of an independent work [5] (placed under the "WF^{2}Q+" class since it is actually an operative version of the WF^{2}Q). The following list provides a bit of information on these WF^{2}Q+ schedulers, which will be analysed more thoroughly later on:

- The S-SPFQ is the only well-defined packet scheduler, in the sense that a real algorithm is provided with it: the system potential approximation comes along with an algorithm which tells when the potential update should be done (this piece of information is needed, as explained in section 2.3).
- The *"Varma" WF^{2}Q+ (V-WF^{2}Q+)* is a WF^{2}Q+ inspired by the S-SPFQ algorithm structure, but with a different set of assumptions; the V-WF^{2}Q+ is defined in this article in order to explore one more direction for the WF^{2}Q operative schedulers, mainly for comparison purposes.
- The *"Zhang" WF^{2}Q+ (Z-WF^{2}Q+)* has been "hinted" at in [4], i.e. some information is missing to get an algorithm out of it (see section 3.2. The Z-WF^{2}Q+ (which should, in Zhang's honour, be referred to as *the* WF^{2}Q+) algorithm can be inferred from some considerations in literature and from a practical implementation of it (the *C-WF^{2}Q+*, a piece of C++ code for the NS 2.1 simulator [8]). But the C-WF^{2}Q+ algorithm seems to be substantially different from Z-WF^{2}Q+. See section 3.2 for further details.

2.2 Approximated System Potential Formulas for the WF^{2}Q+

In order to overcome the calculation of the Virtual Time function, which is a very heavy duty in a real-time implementation, Zhang proposed [4] a Virtual Time function approximation, which is recursively updated on the basis of the overall service amount carried out by the *real scheduler* [3]:

$$V_S(t_2) = max \left\{ V_S(t_1) + W(t_1, t_2), \min_{i \in B(t_2)} SP_i^{h_i(t_2)} \right\} , \tag{2}$$

[3] Another approximated Virtual Time function appears in [5] (where the S-SPFQ is defined), with a different notation: "Potential" (P) instead of "Virtual Time" (V), "Starting, Finishing Potential" (SP, FP) instead of "Starting, Finishing Time" (S, F). In this article these basically equivalent concepts will be used indifferently.

where $W(t_1, t_2)$ is the amount of work carried out by the scheduler S in $[t_1, t_2]$. If the scheduler is constantly active in such a period, $W(t_1, t_2) = t_2 - t_1$. $B(t_2)$ is the set of backlogged connections at t_2 and $h_i(t_2)$ is the index of the head-of-line packet of connection i at time t_2: i.e. $min_{i \in B(t_2)} SP_i^{h_i(t_2)}$ is the smallest starting potential at time t_2. As can be noticed, the potential at a given time is obtained from the potential at a previous time and from the work carried out by the real scheduler in between.

Another layout for the above formula exists [7], and might seem different:

$$\widetilde{V}_S(t_2) = max \left\{ \widetilde{V}_S(t_1) + W(t_1, t_2), \min_{i \in B(t_1)} SP_i^{h_i(t_1)} \right\} . \tag{3}$$

The difference is that the minimum SP is evaluated at time t_1 instead of t_2. The two formulas can be considered the same if $t_1, t_2 \in]t_s, t_e]$ (a single "service interval"), where t_s is the starting time of the current service and t_e is the finish time of the current service. In fact, the minimum SP does not change in $]t_s, t_e]$, as explained in the proof of theorem 2.

2.3 "Transitivity" of Zhang's WF^{2}Q+ System Potential Formula

Zhang's formula allows to evaluate the potential at time t by knowing the potential at a previous time t_0, the work done by the scheduler in $]t_0, t[$ and the minimum SP among all the backlogged queues at time t (briefly indicated as $SP_{min}(t)$). Thus, the $V(t)$ function can be written as:

$$V(t) = f\left(V(t_0), W(t_0, t), SP_{min}(t)\right) . \tag{4}$$

Theorem 1. *With the above potential definition, given t_0, t_1, t_2 belonging to the same service interval, the potential value at time t_2 obtained from the status at time t_0 is different from the potential calculated starting from the status at time t_1, being the potential at time t_1 obtained by the status at time t_0. That is, defining*

$$V(t_2) = f\left(V(t_0), W(t_0, t_2), SP_{min}(t_2)\right) , \tag{5}$$

$$\widehat{V}(t_2) = f\left(V(t_1), W(t_1, t_2), SP_{min}(t_2)\right) , \tag{6}$$

being $V(t_1) = f(V(t_0), W(t_0, t_1), SP_{min}(t_1))$,

$$\exists\, t_0, t_1, t_2 \in]t_s, t_e] \quad | \quad V(t_2) \neq \widehat{V}(t_2) . \tag{7}$$

Proof. Let's write explicitly the formulas:

$$V(t_2) = max\left\{V(t_0) + W(t_0, t_2), SP_{min}(t_2)\right\} ; \tag{8}$$

$$\begin{aligned}\widehat{V}(t_2) &= max\,\{V(t_1)+W(t_1,t_2), SP_{min}(t_2)\} = \\ &= max\,\{max\,\{V(t_0)+W(t_0,t_1), SP_{min}(t_1)\}+W(t_1,t_2), SP_{min}(t_2)\}\ . \end{aligned} \tag{9}$$

$SP_{min}(t)$ is constant $\forall t \in]t_s, t_e]$, since the set of head-of-line packets at t_s does not change and any newly arrived packet on an empty queue (which causes a pool change) will be assigned an SP greater than the $min(SP)$. Let's call K this constant value: $SP_{min}(t_1) = SP_{min}(t_2) = K$; thus,

$$\begin{aligned}\widehat{V}(t_2) = max\,\{max\,\{V(t_0)+W(t_0,t_1), K\}+W(t_1,t_2), K\} = \\ = max\,\{V(t_0)+W(t_0,t_1), K\}+W(t_1,t_2)\ . \end{aligned} \tag{10}$$

Now, there can be two sub-cases:

1. if $V(t_0)+W(t_0,t_1) \geq K$:

$$\widehat{V}(t_2) = V(t_0)+W(t_0,t_1)+W(t_1,t_2) = V(t_0)+W(t_0,t_2) \tag{11}$$

(note that $\forall t_a, t_b, t_c\ \ W(t_a,t_b)+W(t_b,t_c) = W(t_a,t_c)$). Then, since $V(t_0)+W(t_0,t_2) \geq K$ (being $W(t_0,t)$ a monotonous non decreasing function in t), we have:

$$V(t_2) = V(t_0)+W(t_0,t_2) \tag{12}$$

thus, in this case, $V(t_2) = \widehat{V}(t_2)$.

2. but, if $V(t_0)+W(t_0,t_1) < K$, the potential values are:

$$\widehat{V}(t_2) = K + W(t_1,t_2) \tag{13}$$

$$V(t_2) = \begin{cases} \text{if } V(t_0)+W(t_0,t_2) \geq K & V(t_0)+W(t_0,t_2) \\ \text{if } V(t_0)+W(t_0,t_2) < K & K \end{cases} < \widehat{V}(t_2) \tag{14}$$

both values are possible for $V(t_2)$, depending on $W(t_1,t_2)$; for both $V(t_2)$ values, $V(t_2) < \widehat{V}(t_2)$.

□

The above proof shows that the potential updating formula is not enough to define a WF^{2}Q+ scheduler, and the $V(t)$ formula should be re-defined as $V(t,t_0)$. The specification of a potential updating algorithm is also needed. Two scheduling algorithms which use such an approximated potential function and update it at different times may be different, but it is not evident that the different potentials induce different output packets ordering. This aspect has been investigated by means of simulations and presented in section 3.

2.4 Overview of the Algorithms

The symbols used are the following:

1. p_i^k is the k^{th} packet of connection i, L_i^k its length and a_i^k its arrival time;
2. SP_i^k and FP_i^k are, respectively, the starting and finishing potentials of p_i^k;
3. t_e is the time at which the last service ended;
4. $V(t_e^+)$ is the potential evaluated at t_e, but after the calculations done at the end-of-service.

A note on the Z-WF^{2}Q+ and C-WF^{2}Q+: the two algorithms differ from the S-SPFQ and V-WF^{2}Q+ models in that the last ones use the SP and FP variables for each packet in the queue system, whereas the first ones use one (SP, FP) couple for the whole connection, thus reducing the complexity.

Furthermore, S-SPFQ and V-WF^{2}Q+ take actions at three events: *(1)* when enqueueing a packet, *(2)* when deciding on the next packet to send and *(3)* when ending the service of a packet. Z-WF^{2}Q+ and C-WF^{2}Q+, instead, are based on an action to be taken when a packet reaches the head of its queue and on an action taken when scheduling the next packet. Anyway, their "two-triggering-events" scheme can be easily mapped into S-SPFQ's and V-WF^{2}Q+'s "three-triggering-events" scheme (on which we based our table), by distinguishing the two possible times when a packet can reach the head of its queue: when being enqueued on an empty queue or when the packet ahead of it is scheduled for service.

The different scheduling algorithms are summarised in table in Figure 1.

3 Commonalities and Differences

A thorough set of comparisons between the WF^{2}Q derivatives is here presented. The comparisons are carried out at theoretical level and with the support of simulations, as well. The simulation tool is a discrete-event simulator ("*sch_sim*") tailored for packet scheduling algorithms, implemented at the Information Engineering Dept. NetLab of University of Pisa. *Sch_sim* processes input traffic traces running a number of scheduling algorithms along with the corresponding GPS system. Two relevant features are the exact implementation of WFQ and WF^{2}Q algorithms and assessment of theoretical properties of an algorithm (e.g. differences between the real scheduler and GPS queues lengths).

3.1 S-SPFQ and V-WF^{2}Q+

The two algorithms share the same "packet selection" and "end-of-service" operations. The difference is in the "packet arrival" operations, where they have two different ways to calculate the SP of the newly arrived packet. Now, the two algorithms use different terms in the second member of the max expression. The V-WF^{2}Q+ uses

$$V_{VWF^2Q+}(a_i^k) = max\left\{V_{VWF^2Q+}(t_s) + a_i^k - t_s, \min_{i\in B(a_i^{k-})} SP_i^{h_i(a_i^{k-})}\right\} , \quad (15)$$

	Packet arrival	Packet selection	End-of-service
S-SPFQ	❑ If the scheduler is serving a packet: $SP_i^k = \max\{FP_i^{k-1}, V(t_S) + a_i^k - t_S\}$ $FP_i^k = SP_i^k + L_i^k / r_i$ ❑ If the scheduler is idle: *reset all the parameters.*	*Choose the packet with lowest FP, among all the head-of-line packets with SP ≤ V(tE+).*	❑ Potential update: $V(t_E) = \max\left\{V(t_S) + L_i^k / r_i, \min_{i \in B(t_E^-)} SP_i^{h_i(t_E^-)}\right\}$
V-WF²Q+	❑ If the scheduler is serving a packet: $SP_i^k = \max\{FP_i^{k-1}, V(a_i^k)\}$ $FP_i^k = SP_i^k + L_i^k / r_i$ ❑ If the scheduler is idle: *reset all the parameters.*	*Choose the packet with lowest FP, among all the head-of-line packets with SP ≤ V(tE+).*	❑ Potential update: $V(t_E) = \max\left\{V(t_S) + L_i^k / r_i, \min_{i \in B(t_E^-)} SP_i^{h_i(t_E^-)}\right\}$
Z-WF²Q+	❑ If the packet finds an empty queue: $SP_i \leftarrow \max\{FP_i, V(a_i^k)\}$ $FP_i \leftarrow SP_i + L_i^k / r_i$	❑ *Choose the head-of-line packet of the connection with lowest FP, among all the connections with SP ≤ V.* ❑ Then: $SP_i \leftarrow FP_i$ $FP_i \leftarrow SP_i + L_i^k / r_i$	
C-WF²Q+	❑ If the packet finds an empty queue: $SP_i \leftarrow \max\{FP_i, V\}$ $FP_i \leftarrow SP_i + L_i^k / r_i$ $V \leftarrow \max\{V, SP_{MIN}(a_i^{k-})\}$	❑ *Choose the head-of-line packet (p_i^h) of the connection with lowest FP, among all the connections with SP ≤ V.* ❑ If there's a packet p_i^{h+1} after dequeueing p_i^h: $SP_i \leftarrow FP_i$ $FP_i \leftarrow SP_i + L_i^{h+1} / r_i$ ❑ In any case, update: $V \leftarrow \max\{V + L_i^h / r, SP_{MIN}(t_s^+)\}$	

Fig. 1. Example of S-SPFQ and V-WF^{2}Q+ system potential evolution

where t_s is the current service start time. Note that SP_{min} is calculated at a_i^{k-}: the newly arrived packet is not taken into account when evaluating the minimum SP, being its SP not yet defined. This is quite reasonable, thus the "-" sign in a_i^{k-} will not be used in the remainder of this document, because $SP_{min}(a_i^{k-})$ will unambiguously indicate the minimum SP evaluated at the packet arrival on a pool of head-of-line packets which does not include the newly arrived packet. It's easy to see that the S-SPFQ formula for the SP calculation is a simplification of the V-WF^{2}Q+'s one; in fact, S-SPFQ uses

$$V_{SSPFQ}(a_i^k) = V_{SSPFQ}(t_s) + a_i^k - t_s \ . \tag{16}$$

Explanation of the S-SPFQ Potential Updating Formula at Packet Arrival. We can set out the problem according to the following theorem.

Theorem 2. *The S-SPFQ and V-WF^{2}Q+ use the same potential formula (15), but, once started a packet's service, the V-WF^{2}Q+ excludes the packet's SP from the pool $B(t)$ over which $min(SP)$ is evaluated, whereas the S-SPFQ keeps its SP in that pool, and extract it just before ending its service and calculating the new potential. Thus, the $min(SP)$ calculated at end-of-service is the same for both schedulers ($\Rightarrow$ the potentials are the same at that time), whereas the ones evaluated at packet arrival times (during a service) are different ($\Rightarrow$ the potentials may be different).*

Proof. Here it will be proved that the S-SPFQ's different management of the served packet's SP leads to the simplified formula (16).

Be $]t_s, t_e]$ the service interval considered; since the packet under service is not included in the V-WF^{2}Q+'s set of packets for the $min(SP)$ evaluation, $SP_{min}(t)$ is constant in $]t_s, t_e]$ [4] and has a point of discontinuity in t_s: $SP_{min}(t_s^-) \neq SP_{min}(t_s^+)$. The $B_{SSPFQ}(t)$ set, on the contrary, includes the packet under service, until t_e^-, that is just a "thick" before its end-of-service time; then, the SSPFQ $SP_{min}(t)$ is constant in $[t_s, t_e[$ and has a point of discontinuity in t_e: $SP_{min}(t_e^-) \leq SP_{min}(t_e^+)$. Of course, a t_e is the "t_s" of the next interval. It is interesting that, for the S-SPFQ, the $min(SP)$ evaluated at the end of the previous service is the same as evaluated at packets arrivals in $[t_s, t_e[$. Let's refer to the $min(SP)$ value in $[t_s, t_e[$ as K:

$$\begin{aligned} V_{SSPFQ}(t_s) &= V_{SSPFQ}(t_{e_{prev}}) = \\ &= max\left\{V_{SSPFQ}(t_{s_{prev}}) + t_{e_{prev}} - t_{s_{prev}}, K\right\} \geq K \ , \end{aligned} \tag{17}$$

where $t_{s_{prev}}$ and $t_{e_{prev}}$ are, respectively, the starting and ending times of the previous service. Note that $t_s = t_{e_{prev}}$, and is the instant soon before the starting of the current service, when the potential is updated (being the set of SPs updated), as explained in the following. Thus, a-fortiori

$$\forall a_i^k \in [t_s, t_e[\quad V_{SSPFQ}(t_s) + a_i^k - t_s \geq SP_{min}(t_s) = SP_{min}(t_s^+) = K \ . \tag{18}$$

□

Explanation of the Notation. A clarification is needed on the notation used to express the end-of-service ($\equiv$ beginning-of-service) times (t_{s,e^-} , $t_{s,e}$, t_{s,e^+}). At these times, both S-SPFQ and V-WF^{2}Q+ perform instantaneously two main changes to the system status: *(1)* the potential function is updated and *(2)* a new packet is scheduled for transmission – and the $min(SP)$ either changes (V-WF^{2}Q+) or not (S-SPFQ). In order to distinguish among the various steps at the same instant $t_{s,e}$, the following notation is used in this article:

- $t_{s,e}^-$ is the time right before the potential update; nothing has changed yet.
- $t_{s,e}$ indicates that the potential has been updated (for the S-SPFQ, this means that the ended packet's SP has been removed from the pool), but a new packet has not been scheduled yet.
- $t_{s,e}^+$ indicates that the new packet has been scheduled; in case of V-WF^{2}Q+, the packet's SP has been removed from the pool over which $min(SP)$ is calculated; everything now has been done.

Thus, $V(t_{s,e}^-)$ is the potential right before the update, $V(t_{s,e})$ is the updated potential and (in the V-WF^{2}Q+ case) $V(t_{s,e}^+)$ is a new potential value, since it is calculated as the maximum between $V(t_{s,e})$ and the (possibly) new $min(SP)$.

[4] For both V-WF^{2}Q+ and S-SPFQ, a new packet arrival on an empty queue in $]t_s, t_e]$ may change the $B(t)$ set but not the $min(SP)$, since the new packet's $SP \geq V(t_s) \geq SP_{min}(t_s)$, and $SP_{min}(t_s)$ belongs to a packet in the system in $]t_s, t_e]$.

Now we can better explain the statement on the $min(SP)$ discontinuities. In both V-WF^{2}Q+ and S-SPFQ, the $min(SP)$ has a discontinuity at the end (beginning) of services. But in the V-WF^{2}Q+ the discontinuity is associated to action 2, i.e. between $t_{s,e}$ and $t^+_{s,e}$, whereas in the S-SPFQ the discontinuity is associated to action 1, i.e. between $t^-_{s,e}$ and $t_{s,e}$.

Temporal Evolution of V-WF^{2}Q+'s and S-SPFQ's System Potential. This section provides a graphical representation example of the qualitative trends of S-SPFQ and V-WF^{2}Q+ system potentials. The right plot in Figure 2 allows to see the two potential overlaid. The parts where $V_{SSPFQ}(t)$ differs from $V_{VWF^2Q+}(t)$ are represented as dashed lines.

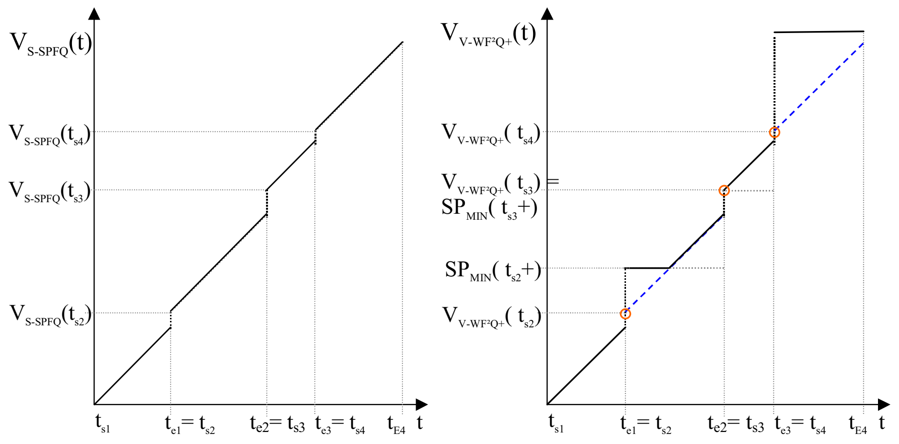


Fig. 2. Example of S-SPFQ and V-WF^{2}Q+ system potential evolution

Difference between V-WF^{2}Q+ and S-SPFQ SPs Evaluated at Packet Arrivals. The different S-SPQF and V-WF^{2}Q+ potential formulas may cause different potential values to be evaluated at a packet arrival (assumed that the server is busy) and, consequently, a different Starting Potential assigned to that packet. A further discussion is needed on the last consequence. First, let's recall that ${}^{S}SP^k_i = max\left\{{}^{S}FP^{k-1}_i, V_S(a^k_i)\right\}$, where "$S$" is either S-SPFQ or V-WF^{2}Q+. Using the formulas (15) and (16), the SPs can be written as:

$$^{VWF^2Q+}SP^k_i = max\left\{^{VWF^2Q+}FP^{k-1}_i, V_{VWF^2Q+}(t_s) + a^k_i - t_s, SP_{min}(a^k_i)\right\} \tag{19}$$

$$^{SSPFQ}SP^k_i = max\left\{^{SSPFQ}FP^{k-1}_i, V_{SSPFQ}(t_s) + a^k_i - t_s\right\} \tag{20}$$

Let's divide our discussion into two cases:

1. If packet arrives at a_i^k and finds an empty queue, and its previous packets was served "enough" time before a_i^k, it may be true that
$$FP_i^{k-1}\ \left(= SP_i^{k-1} + L_i^{k-1}/r_i\right) < V_S(t_s^+)\ \left(\leq V_{SSPFQ}(a_i^k) \leq V_{VWF^2Q+}(a_i^k)\right) \tag{21}$$
this is true if SP_i^{k-1} is lower "enough" than the $min(SP)$ which contributes to $V_S(t_s^+)$ (i.e. if p_i^{k-1} was served enough time ago). Here we used the property that the SP of a packet is always higher than the SP of a previously arrived packet. If, in addition, $SP_{min}(a_i^k) \leq V_{VWF^2Q+}(t_s) + a_i^k - t_s$, we get:
$$^{VWF^2Q+}SP_i^k =^{SSPFQ} SP_i^k = V_S(t_s) + a_i^k - t_s \tag{22}$$
(remember that V-WF^{2}Q+'s and S-SPFQ's potentials have the same value right after the beginning of a service). This, actually, proves that the SPs assigned to a packet by V-WF^{2}Q+ and S-SPFQ may be equal.
2. But, in any other case, e.g. when a packet enters a backlogged session, the above assumptions can't be made, and the new SP may be equal either to the previous packet's FP (which may be different for the two schedulers) or to system potential (which may be different for the two schedulers).

Are the V-WF^{2}Q+ and S-SPFQ Different? The possibility that different SPs are assigned by the V-WF^{2}Q+ and S-SPFQ to a newly arrived packet, may not prevent the two schedulers from having always the same output packets sequence (thus being equivalent systems).

The difference between the V-WF^{2}Q+ and the S-SPFQ systems can be proved with just a single example where the two schedulers produce different output sequences under the same input sequence; thus, the proof has been found using the *sch_sim* simulator, configured with different scenarios. In some cases, the two schedulers actually resulted in different outputs. Here in the following, one of these situations is described.

– General configuration:

```
input from '2_video_flows', output into 'wf2qplus.outseq.1',
duration= 60 sec, link rate = 250 Kb/s, 2 queues:
Q[1]:    maxlength = 100000 B,  service rate = 170 Kb/s
Q[2]:    maxlength = 100000 B,  service rate = 80 Kb/s
```

– The output of the two systems has been exactly the same until $t = 308.91$ ms. At this time the situation is the following:
  - connection 1 has a queue level = 0 bytes;
  - connection 2 has a queue level = 3331 bytes; the head-of-line packet has SP=314.606 ms and **FP=352.006 ms** and a packet under service (FP = 314.606 ms), started at t = 297.293 ms and finishing at t = 309.261 ms.

- then, at $t = 308.91$ ms, a new packet (1064 bytes) arrives at queue 1.
  - the V-WF^{2}Q+ assigns the following values:

```
ENQUEUE: q[1]: t=308.910000, size=1064, qlen_found=0,
         pkt=0x58738, pot=277.206000(+T=288.823000),
         sp=314.606000, fp=364.676588 (prev=269.080353),
          min_sp=314.606000[2,0x58408], pot_used=314.60600}
```

  - the S-SPFQ assigns the following values:

```
ENQUEUE: q[1]: t=308.910000, size=1064, qlen_found=0,
         pkt=0x58738, pot=277.206000, sp=288.823000,
         fp=338.893588 (prev=269.080353), pot_used=288.8230
```

- at $t = 309.261$ ms, the current service ends and a new packet is scheduled:
  - the V-WF^{2}Q+ updates the potential:

```
EOS    : Q[2]: t=309.261000, pkt=0x584e0, gps_pkt=0x58c50
         pot=314.606000, min_sp=314.606000[1, 0x58788]
```

    and makes the following choice:

```
SERVNXT: t=309.261000
         Q[2]: pkt=0x58458, size=374, min_fp=352.006000,
         pot=314.606000
```

  - the S-SPFQ updates the potential:

```
EOS    : Q[2]: t=309.261000, pkt=0x584e0, gps_pkt=0x58c50
         pot=289.174000, min_sp=288.823000[1, 0x58788]
```

    and makes the following choice:

```
SERVNXT: t=309.261000
         Q[1]: pkt=0x58788, size=1064, min_fp=338.893588,
         pot=289.174000
```

That is, in this case, the two schedulers choose different packets for transmission. *Thus, we proved that at least one case exists where V-WF^{2}Q+'s and S-SPFQ's different SPs (and FPs) cause a different scheduling decision.*

3.2 Z-WF^{2}Q+ and C-WF^{2}Q+

Some notes are required on the two algorithms before beginning a comparison:

- In the Z-WF^{2}Q+, the potential updating times are not specified; the potential used to calculate the SP of a packet which reaches the head of an empty queue should be evaluated at the arrival time of the packet (see Figure 1). But, as shown in section 2.2, the potential formula is not "transitive", and the potential value at a given time depends on the updating algorithm. Thus, saying "use V evaluated at the arrival time of the packet" is not sufficient.

- The C-WF^{2}Q+ updates V when enqueueing a packet on an empty queue. To this purpose, the minimum SP is calculated just before enqueueing the packet: this is the meaning of the "$-$" sign in a_i^{k-}.
- The C-WF^{2}Q+ updates V after scheduling a packet for service. In the calculation of the minimum SP at this time (t_s), the scheduled packet is considered already removed from its queue: this is the meaning of the "+" sign after t_s^+.

Discussion on the Z-WF^{2}Q+ and C-WF^{2}Q+. The comparison between Z-WF^{2}Q+ and C-WF^{2}Q+, is made difficult by the fact that *the Z-WF^{2}Q+ is not a completely specified algorithm* (in the sense reported above).

Here the C-WF^{2}Q+ is assumed to be the "official" implementation of the Z-WF^{2}Q+ scheduler, and its potential updating policy will be inspected. In fact, the C-WF^{2}Q+ appears to be using, at each event, the potential value updated at the end of the previous event, and this may result in a unclear evaluation of the potential used in the SP calculation.

As can be noticed, then, the potential updated at the beginning of a packet's service is anticipated with respect to its real value: in fact, it is immediately added with the packet's service time (i.e. Zhang's formula is not applied). Then, when a packet is enqueued, the potential is updated, but it is not added with the amount of service done by the scheduler since last event (i.e. Zhang's formula is not applied), probably due to the fact that the whole service done for the packet under transmission has already been taken into account.

A simple example is enough to show that the C-WF^{2}Q+ does not use the "correct" potential value when enqueueing a packet (on an empty queue):

- initially the whole system is idle (no packets): the potential is zero;
- at t_1 a packet arrives at connection i and is served;
- at t_2 a second packet arrives at connection j, with the previous packet still under service.
- Now, the real potential value should be $t_2 - t_1$ (and this also should be the corresponding GPS potential value), but the C-WF^{2}Q+ already increased V of the whole service time of the first packet at its beginning-of-service time: thus, V is higher in C-WF^{2}Q+ than it should.

Thus, this example shows that the C-WF^{2}Q+ potential may be far from the GPS potential.

Definition of a Z-WF^{2}Q+ Algorithm. As explained before, the potential value at a given time is defined by both an updating formula and an updating algorithm (which specifies when the potential value has to be updated). The comparison between the C-WF^{2}Q+ and the Z-WF^{2}Q+ needs some further assumptions on the latter in order to be completed.

In [6], Varma says that "In an idelized fluid server it is possible to update the system potential at any instant of time. However, in a packet-by-packet server it is desirable to update the system potential only when a packet departs the system." This choice unambiguously defines an updating policy and, consequently, the time evolution of the system potential.

The following algorithm is the result of the above consideration applied to the Z-WF^{2}Q+:

- *packet arrival on an empty queue*: $SP_i \leftarrow max\{FP_i, V(a_i^k)\}$ and $FP_i \leftarrow SP_i + L_i^k/r_i$;
- *packet scheduling*: if a service just ended, update the potential [5] and choose the head-of-line packet of the lowest FP connection among all the connections with $SP \leq V$; then update: $SP_i \leftarrow FP_i$ and $FP_i \leftarrow SP_i + L_i^k/r_i$.

A set of simulations carried out on *sch_sim* showed that the C-WF^{2}Q+ and this Z-WF^{2}Q+ (with potential update at end-of-service) produce different output packet sequences (i.e. are different algorithms).

3.3 Z-WF^{2}Q+ and V-WF^{2}Q+ / S-SPFQ

The most evident difference between the V-WF^{2}Q +/S-SPFQ and Z-WF^{2}Q+ (as previously defined) approaches is the association of SPs and FPs with the single packet in the former cases, and with the whole connection in the latter. In this section we will show that such a difference leads to a different output.

The Z-WF^{2}Q+ algorithm says that the SP and FP of the connection have to be updated when a packet reaches the head of the queue:

- if the packet arrived at an empty queue: $SP_i \leftarrow max\{FP_i, V(a_i^k)\}$;
- if the packet found at least a packet ahead at its arrival: $SP_i \leftarrow FP_i$.

This is equivalent to update SP as soon as a packet arrives and to bind its SP with it:

- if the packet arrived at an empty queue: $SP_i^k = max\{FP_i^{k-1}, V(a_i^k)\}$;
- if the packet found at least a packet ahead at its arrival: $SP_i^k = FP_i^{k-1}$.

The "empty queue" case formula is the same as in the V-WF^{2}Q+ and S-SPFQ. The second case formula, instead, is based on the assumption that $FP_i^{k-1} \geq V(a_i^k)$, where $V(a_i^k) = max\{V(t_s) + a_i^k - t_s, SP_{min}(a_i^k)\}$. Now, since p_i^{k-1} was still in queue at a_i^k, it is true that $FP_i^{k-1} \geq SP_i^{k-1} \geq SP_{min}(a_i^k)$; but we cannot say that $FP_i^{k-1} \geq V(t_s) + a_i^k - t_s$.

Simulations showed that some cases exist where the Z-WF^{2}Q+ produces a different output with respect to either V-WF^{2}Q+ and S-SPFQ.

3.4 C-WF^{2}Q+ and V-WF^{2}Q+ / S-SPFQ

The same discussion applied previously when showing the difference between Z-WF^{2}Q+ and C-WF^{2}Q+ can be now used to explain the difference between the C-WF^{2}Q+ and both the V-WF^{2}Q+ and S-SPFQ. Simulation results confirm this.

[5] $V \leftarrow max\left\{V + L_i^{k-1}/r_i, min_{i \in B(t_e^-)} SP_i^{h_i(t_e^-)}\right\}$

3.5 Final Considerations on the Comparisons

The above comparisons showed that S-SPFQ, V-WF^{2}Q+, Z-WF^{2}Q+ and C-WF^{2}Q+ are all different algorithms, in the sense that they may produce different packet sequences under some circumstances. This may not prevent them from having equivalent performances, as is discussed in the following.

4 WF^{2}Q+ Schedulers' Compliance with WF^{2}Q Bounds

Another set of simulations showed that each implementation of a WF^{2}Q+ algorithm is different from the original WF^{2}Q (producing a different packet sequence). What is relevant, then, is to check if the WF^{2}Q+ implementations still fulfill the WF^{2}Q fairness property [6]. This property comes out from the following *sufficient* conditions characterising the WF^{2}Q:

$$\forall i,k \quad d^k_{i,WF^2Q} - d^k_{i,GPS} \le L^{MAX}/r \; , \tag{23}$$

$$\forall \tau \quad Q_{i,GPS}(\tau) - Q_{i,WF^2Q}(\tau) \le (1 - r_i/r)L_i^{MAX} \; , \tag{24}$$

and from the GPS property: $\forall i,k \;\; d^k_{i,GPS} - a^k_i \le Q_{i,GPS}(a^k_i)/r_i$.

The simulations showed that for each WF^{2}Q+ implementation some cases occur where one of the first two (or both) conditions are *not* satisfied. Here is a sample of such simulations on the C-WF^{2}Q+; similar results apply to the other WF^{2}Q+ schedulers [7]:

```
sch_sim:simulation parameters:
        input from '2_video_flows', output into 'cwf2q+.outseq.1',
        duration= 60 sec
        scheduler = 'c-wf2q+'(40), link rate = 250 Kb/s, 2 queues:
        Q[1]:    maxlength = 100000 B,  service rate = 170 Kb/s
        Q[2]:    maxlength = 100000 B,  service rate = 80 Kb/s

Results:
Q[1]:   servc:  sent: 1223786 B, 2290 pkts; recv: 1223786 B,
                2290 pkts; drops: 0 pkts, avg_rate= 163.17 Kb/s
        delays: avg= 320.36 ms, max= 1117.21 ms,
                wfi= 45.783 ms (th:50.353 [ok]),
                Dsch-Dgps_max= 48.473 ms (th:34.240 [no])
        queues: max_real= 23634 B, max_gps= 23490.3 B,
                sched-gps_min= -799.4 B (th:-342.4 [no]),
                sched-gps_max= 1149.7 B (th:1070 [no])
```

[6] Each connection WFI should be lower than the WF^{2}Q bound defined in [1].

[7] For each listed connection i, `sched-gps_min` is $min\{Q_{i,WF^2Q}(\tau) - Q_{i,GPS}(\tau)\}$ for $\tau \in$ the simulation interval, and `Dsch-Dgps` is the $max_k\{d^k_{i,WF^2Q} - d^k_{i,GPS}\}$ for the connection. Each parameter is followed by its theoretical value ("`th`").

```
Q[2]:   servc:  sent: 651214 B, 1682 pkts; recv: 998258 B,
                2023 pkts; drops: 341 pkts, avg_rate= 86.82 Kb/s
        delays: avg= 7466.13 ms, max= 9981.71 ms,
                wfi= 58.648 ms (th:107.000 [ok]),
                Dsch-Dgps_max= 61.141 ms (th:34.240 [no])
        queues: max_real= 99990 B, max_gps= 99999.2 B,
                sched-gps_min= -1149.7 B (th:-727.6 [no]),
                sched-gps_max= 799.4 B (th:1070 [ok])
```

Obviously, this does not prevent the fairness theoretical bound from being fulfilled by the WF^{2}Q+ schedulers; in fact, all the simulations showed that such a bound is still true for the Z-,V-,C-WF^{2}Q+ and for the S-SPFQ, as well.

This leads to conclude that, notwithstanding the differences among the four WF^{2}Q+ algorithms, they still perform within the theoretical bound and can be used indifferently when a WF^{2}Q level of fairness is required.

5 Conclusions

This article summarises the work done around the "most fair" WF^{2}Q scheduler, discussing at both theoretical and simulation level the differences among its WF^{2}Q+ implementations. Some of them are already clearly defined (S-SPFQ and C-WF^{2}Q+), whereas others are "interpolated" from basic ideas and proposals in literature. At the end of the analysis process, each one of the WF^{2}Q+ implementation showed to be different from the others, since the produced output packet sequences differ in some cases. These differences, however, does not affect their performance equivalence (in terms of delay and fairness bounds).

Acknowledgements: This work was partially carried out with the support of the Italian Ministry of University and Scientific Research (MURST Contract N. 9809321920) "Techniques for quality of service guarantees in multiservice telecommunication networks".

References

1. J.C. Bennet, H. Zhang: WF^{2}Q: Worst-case Fair Weighted Fair Queueing. INFOCOM '96 (1996)
2. A. Demers, S. Keshav, S. Shenkar: Analysis and Simulation of a Fair Queueing Algorithm. Internet. Res. and Exper., vol. 1 (1990)
3. A.K. Parekh, R. Gallager: A Generalized Processor Sharing Approach to Flow Control in Integrated Services Networks: The Single-Node Case. IEEE/ACM Trans. on Networking, Vol. 1 , No. 3 (1993)
4. J.C. Bennet, H. Zhang: Hierarchical Packet Fair Queueing Algorithms. IEEE/ACM Trans. on Networking, Vol. 5, No. 5 (1997)
5. A. Varma, D. Stiliadis: Hardware Implementation of Fair Queueing Algorithms for Asynchronous Transfer Mode Networks. IEEE Communication Magazine (1997)

6. D. Stiliadis, A. Varma: A General Methodology for Designing Efficient Traffic Scheduling and Shaping Algorithms. IEEE Infocom '97 proceedings (1997)
7. J.C.R. Bennet, D. C. Stephens, H. Zhang: High Speed, Scalable and Accurate Implementation of Packet Fair Queueing Algorithms in ATM Networks. IEEE Communication Magazine (1997)
8. http://www.cs.cmu.edu/~cheeko/wf2q+/

Requirements on the TCP/IP Protocol Stack for Real-Time Communication in Wireless Environments

Lars-Åke Larzon[1], Mikael Degermark[1,2], and Stephen Pink[1,2]

[1] Dept of CS&EE, Luleå University of Technology, Luleå, Sweden
[2] Dept of CS&EE, University of Arizona

Abstract. A strong trend in the Internet is to provide wireless Internet access to handheld devices. Radio spectrum is a scarce and expensive resource, and a wide-area cellular system that provides satisfying quality for a data service based on the traditional TCP/IP protocol stack will be expensive. This becomes especially problematic for real-time communication based on UDP.

To design an economically viable system, different approaches are available. New protocols can be designed that take wireless systems into account. The Wireless Application Protocol (WAP) is an example of such a protocol. Alternatively, the existing TCP/IP protocols can be modified to better support cellular systems. The development of WAP is tempting since it does not have to modify the TCP/IP Protocol stack. However, to provide future proofing, it is a better idea to base the protocol architecture for wireless on TCP/IP since this will be the dominant Internet protocol stack for years to come.

This paper discusses requirements on various layers in the TCP/IP protocol stack to better support real-time data services. One key component is a transport layer designed to use limited bandwidth more efficiently for real-time services. Changes are required to other layers as well, but these can be deployed where needed without disturbing the rest of the Internet. By applying these changes, it becomes possible to provide cost efficient, real-time services based on IP in cellular systems.

1 Introduction

With the advent of IP telephony running atop cellular networks, new problems arise for the traditional Internet protocols. In cellular systems, the scarce - and most expensive - resource is the radio spectrum. To provide customer services at reasonable prices, the spectrum must be used as cost-efficiently as possible. As a result, cellular systems typically have occasionally high variations of relatively high delay, low bandwidth and frequent bit errors. To overcome the problems with these properties, services in cellular networks typically avoid retransmissions, minimize protocol overhead and use as much as possible of the data that reach the destination. Internet protocols, on the other hand, often use retransmissions, have spacious headers and typically discard entire packets if a single bit error is detected. As a result of this, traditional Internet protocols often exhibit bad performance in cellular networks.

From an economical perspective, it is attractive for operators and service providers to use Internet protocols as the foundation of all communication, but the performance is unsatisfying. There are basically three different kinds of solutions to these problems:

1. Modify the characteristics of the cellular networks
2. Modify the appropriate Internet protocols
3. Define a new and better set of protocols.

Modifying the characteristics of cellular networks can be done to some extent at the cost of not being able to use the radio spectrum efficiently. As a result, not as many users can use the system simultaneously. To compensate for reduced user capacity, operators and service providers must increase their prices. Besides resulting in a system that is more expensive to end users, reducing the number of possible simultaneous users in a cell is in conflict with the rapidly increasing number of users.

Changing the Internet protocols involved requires careful design to preserve compatibility with hundreds of millions of existing Internet users. Given the size of the Internet, there is also a significant deployment problem if new protocols are required to be installed in a large part of the Internet.

Defining a new set of protocols provides an opportunity to improve on the existing Internet stack. However, with the huge deployment base of Internet protocols, it would take too long for a new protocol suite to get sufficient acceptance to provide the same connectivity as the current Internet.

What is desired is to provide cost-efficient cellular networks with Internet connectivity that enable services with sufficient quality. If a combination of the first two alternatives is chosen, it might be possible to redesign or introduce new protocols in a transparent and backward-compatible way which better conforms with the properties of cellular networks. With such modifications, cellular systems might not have to reduce spectrum in order to provide satisfactory service quality.

This paper discuss what requirements must be made on the Internet protocol stack in order to work better in heterogeneous networks in general, and cellular networks in particular. After outlining these requirements, a set of suggested solutions are proposed together with issues for further investigation.

In section 2, we present requirements on the protocol stack. Suggested solutions follow in section 3, and a discussion in section 4. Finally, some conclusions and notes about current and future work.

2 Requirements

In a wide-area wireless network, the radio spectrum is a scarce, and also very expensive[1], resource. For their users, operators want to provide an attractive service that has acceptable quality at a reasonable cost. Providing sufficient service quality in a wireless network comes down to keeping the delay and bit error rates (BER) at an acceptable level, without sacrificing too much of the valuable radio spectrum.

Consider as an example the GSM cellular phone network. The BER after channel coding is in the range of 10^{-3}-10^{-5}, which is quite high. However, the wireless bandwidth is efficiently used by error-tolerant speech codecs and a radio channel capable of delivering damaged speech data provided the damage is not too severe. Introducing a data service based on the TCP/IP protocol suite in such an environment gives bad performance since damaged data is considered unusable. Moreover, the headers in Internet protocols use a significant part of the available bandwidth.

In this section, we identify requirements on the entire system as well as on the layers in the TCP/IP protocol stack. These requirements should be met to better support real-time IP applications in cellular systems. We focus on real-time applications such as voice over IP since that has been identified as key future application.

2.1 System Requirements

The main problem with the TCP/IP protocol suite is that it is, by and large, designed for traditional data applications using links with relatively low delay, low BER and inexpensive bandwidth. It is not surprising that a cellular network whose channels can have high delay, high BER and low bandwidth will cause significant problems. A data service must be designed to use the limited radio spectrum efficiently. Achieving this comes down to the following three tasks:

- Reducing the impact of delay-sensitive protocol mechanisms to compensate for high delay
- Reducing protocol overhead to compensate for limited bandwidth
- Avoiding discarding usable data to compensate for high BER

These goals can be achieved in various ways. A new protocol stack could be designed, taking the properties of wireless systems into account.[2] An alternative approach is to modify the existing TCP/IP protocol stack to better support wireless systems. This is a more attractive solution since TCP/IP is already widely spread to hundreds of millions of users providing global Connectivity. This can not be done without modifying the existing TCP/IP protocol stack. A

[1] For an operator providing a wireless service, the cost of the radio spectrum can be as high as 1/3 of the total cost.

[2] This approach is chosen in the Wireless Application Protocol (WAP) [13].

discussion of what the requirements on the protocol stack are to provide a data service for wireless environments is the topic of the next sections.

In addition to the requirements discussed in the following sections, there are a few general requirements on the Internet protocols when modifying them for wireless systems. First, the solutions should be deployed locally, to the wireless networks only. Second, locally implemented solutions should be transparent to the rest of the (wired) network, as well as end applications.

2.2 Application Layer

In order to compensate for heterogeneous network paths with variations in delay, bandwidth and error rates, applications should not make any assumptions about network properties. Applications with tight delay restrictions, e.g., interactive voice, can avoid most problems by using UDP in combination with loss and error-tolerant media codecs. If nothing is known about the bottleneck bandwidth between two endpoints, the assumption should be that it is low.

In order to make full use of error-tolerant media codecs, underlying protocols must not discard packets containing usable data. Assuming there exist mechanisms in the underlying layers to prevent this from happening, where is the specification of what errors are acceptable best made? The sending application usually has the best knowledge about this, and therefore it can be argued that it is best suited to specify what parts of the data are to be protected. We hereafter call data where errors are unacceptable *sensitive data*, while data which might be usable despite bit errors in it are *insensitive*.

On the receiving side, the application might benefit from knowing what data was deemed sensitive and thus protected. Insensitive data may need to be verified and possibly discarded if damage is too severe. This will introduce extra complexity in applications at the gain of not having to compensate for packets discarded by underlying protocols due to errors in insensitive data.

2.3 Transport Layer

For real-time applications with tight delay bounds, the unreliable datagram delivery service of UDP is typically used in the Internet. The main problem with UDP, however, is the checksumming policy. Single bit errors that are detected with a failing checksum result in packet loss. This conforms badly with the concept of insensitive data and codecs in the application layer that can handle errors to some extent. In order to reduce this problem, the transport layer should be able to treat sensitive and insensitive data differently.

Since an application should only need to communicate with the transport layer, the transport layer should provide a way for the application to specify the sensitive parts of a packet. This information must be sent along with the packets to the transport layer at the receiving side. For incoming packets, errors in insensitive data must be ignored since the packet still might be usable to the application.

Modifications in the protocol design to achieve handling of insensitive data should be kept at a minimum to simplify protocol implementation and reduce overhead. Preferrably, the modifications should - if possible - be backwardly compatible to simplify deployment.

The transport layer should not by itself decide whether some part of a packet is sensitive or not - this decision is best made by the sending application. Information about data sensitivity is best placed in protocol headers to avoid out-of-band signaling, which is likely to increase both the delay and complexity of the transport protocol.

2.4 Network Layer

Global connectivity in the Internet is provided by the Internet Protocol (IP). Version 4 of IP (IPv4)[12] is still dominant, but version 6 (IPv6)[2] is coming. As for the transport layer, the network layer should not discard packets due to errors in insensitive data. It must also ensure that the information about data sensitivity is given in each packet delivered downwards in the protocol stack. Since the IP checksum does not cover anything but its header, it does not itself have to take insensitive data into account.[3]

If fragmentation can occur in the network layer, information about data sensitivity must be passed in each fragment to allow underlying layers to treat each fragment correctly. If encryption on the network layer level is used, this must be tolerant to errors in insensitive data. The most common assumption when encryption is used is that damaged data has been tampered with. In a cellular system that implements the concept of insensitive data, this is clearly not the case.

2.5 Link Layer

There are many types of link layers with different error characteristics and error protecting mechanisms. Most of these use a checksum-based verification mechanism. A common feature among link layers with high BER is to have mechanisms to optionally treat some part of the frames as insensitive to errors.

Assuming that the requirements of the previous subsections are met, the link layer should use the information from the network layer about insensitive data so that frame loss due to transmission errors in such data is avoided as far as possible. This is most important for link layers with high BERs. Link layers for wired high-speed networks with low BER typically do not have the ability to have different protection levels, and the gain of introducing such mechanisms would be negligible. Such link layers should not be modified.

2.6 Header Compression

One method to reduce the protocol overhead in networks with limited bandwidth is to use header compression to reduce the protocol overhead. Using such

[3] IP headers are assumed to be sensitive data.

compression, headers of up to 60 bytes can be compressed roughly by a factor of 10. If a header compression algorithm is used, this must ensure that the sensitivity information specified by the sending application in the protocol headers is correctly reconstructed at the decompressor.

3 Solutions

In this section, we present existing and suggested solutions to the requirements from the previous section.

3.1 Application Layer

There exists several algorithms for speech and video coding capable of handling a damaged payload. When implemented to use the TCP/IP protocol stack, these algorithms might not expect damaged packets and therefore not implement all functionality. This must be corrected.

Some algorithms might also find it usable to know what data were deemed insensitive. Passing this information from the transport layer to the application is doable by modifying the network API.

3.2 Transport Layer

The UDP Lite protocol[10] meets the requirements on the transport layer in the previous section. It is a small modification of the original UDP protocol. UDP Lite replaces the redundant length information (this information is already part of the IP header) in the UDP header with a Coverage value as shown in figure 1. The Coverage value is equal to or less than the UDP Lite datagram length and specifies how many bytes from the beginning of the UDP Lite datagram are covered by the UDP Lite checksum. If Coverage is equal to the datagram length, the UDP Lite datagram looks and behaves exactly like a classic UDP datagram. With a Coverage less than the UDP Lite datagram length, the last part of the payload will not be verified and errors in that part will consequently be ignored by the transport layer. The checksum coverage must be at least 8 bytes of the packet , which means that at least the pseudo header and the UDP Lite header are protected.

By placing sensitive data in the beginning of each packet and using the optional partial checksum provided by UDP Lite, a real-time application in a wireless environment with properties that make use of classic UDP virtually impossible, can reduce the number of lost packets significantly. Simulations and emulations have shown that using UDP Lite instead of classic UDP in a cellular network makes an IP-based telephony service feasible[9].

The sensitivity of each packet is reflected in the coverage value, which can be specified on a per-packet basis. UDP Lite is a transport protocol with the same low delay and lack of protocol complexity as classic UDP. It provides have the option to use a partial checksum and get better performance for error-tolerant

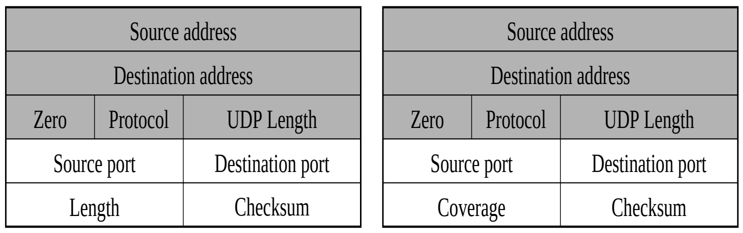


Fig. 1. The UDP and UDP Lite headers, respectively

applications in wireless environments where classic UDP is simply not good enough.

3.3 Network Layer

Currently, IP has no support for sending information about insensitive data up- or downwards in the protocol stack. It can be argued that for IP packets containing the transport layer headers, such an indication is not necessary for outgoing packets since lower layers can peek at transport layer headers. This, however, will mean that lower layers must know the semantics of the transport layer; a clear layer violation. It would be better if the IP headers could include this indication in some way. This can be achieved in different ways.

The most straightforward solution for IPv4 would be an IP option indicating partially insensitive data and an offset to this data within the packet (assuming that all data from the offset until the end of the packet are considered insensitive). A disadvantage of a solution like this is that such packets will use the slow path in most routers along a network path, which might increase the end-to-end delay. On the other hand, an advantage is that this solution is general enough to work with most types of IP packets regardless of the transport protocol. Indication of insensitive data can also occur in packet fragments where transport layer headers are not available.

If it is acceptable for lower layers to peek at the transport layer, there will still be a problem with packet fragments. This can be solved by using one bit in the IP header indicating whether fragments are sensitive to errors or not. Such a solution implies that fragments must be either totally sensitive or insensitive and that the IP layer must take this into account when doing fragmentation.

For IPv6, information about insensitive data can be carried in an extenstion header, which is a solution similar to having a new IP option in IPv4. Alternatively, the IPv6 fragment header has a sufficiently large number of reserved bits available to provide the information. The same arguments for and against as described in the two previous paragraphs will apply also to this solution.

The IPsec protocol[8] does not support insensitive data today since damaged data is considered tampered with. As a direct consequence of this, data security has to be implemented above the network layer. Given the increasing number of wireless nodes in the Internet and the fact that it might be desirable to deliver some damaged data to applications on such nodes, a security scheme should be able to handle insensitive data. We do not suggest how to achieve this in this paper, but we consider it an important issue for further investigation.

3.4 The Link Layer

While it is desirable to make the link layer use information about insensitive data in packets coming from upper layers, it is important to note that we do not suggest changes to existing and widely spread link layer protocols. We do, however, believe that it is important to consider the possible occurrence of insensitive data when designing link layer protocols for future networks. It is also important to note that not all link layers have to take this into Account. In a wired, high-speed environment with a low BER (such as Ethernet, Token Ring or similar), this issue is not important since errors are relatively rare.

Since it is desirable to use whatever is usable of the information reaching the destination over a wireless link, link layers in these networks do have support for insensitive data. In a cellular network, it is common to have the data partioned in different blocks protected by checksums and error recovery mechanisms of various strengths. For a data service based on TCP/IP, these mechanisms are not used since it is impossible to know what errors are acceptable. With a network layer providing the link layer with this information, the mechanisms can provide better data service.

3.5 Header Compression

There are several header compression algorithms for compression of TCP/IP[6,4], UDP/IP[4] and RTP/UDP/IP[1] headers. In addition to this, there are new header compression schemes for RTP/UDP/IP designed for low-bandwidth wireless systems[7,11,5] to support IP-based telephony services over these types of networks. The motivation for these new algorithms is that compressed RTP (CRTP) does not give satisfying performance in systems with high error rates and a round-trip time much greater than the sample time.

Using such header compression algorithms will meet the requirement of reducing protocol overhead, but to get best performance header compression must be combined with a protocol stack that prevents discarding of useable data. It has been shown that header compression performs significantly better when only the header is protected by a checksum, and not the payload. The reason is that more headers are then delivered to the decompressor which is then less likely to lose synchronization with the compressor[3].

4 Discussion

There are several existing solutions for providing data services with Internet access to mobile users. The only one which can be called a widespread standard is the Wireless Application Protocol (WAP) which provides a protocol stack designed especially for wireless environments.

The difference between WAP and the solution we outline in this paper is that our solution is based on the already widespread TCP/IP protocol stack. Since TCP/IP is likely to exist in some form in the devices mentioned, it is cost-efficient to base all services on TCP/IP. Basing the communication on a protocol that is likely to be the dominant in Internet communications for several years ahead, should be future-proof as well as providing higher connectivity.

The solutions we have outlined can be introduced in an Internet Protocol stack and used to communicate with a remote node on the other side of the Internet using the same stack. Some functionality can be deployed without modifying any intermediate nodes except the access points closest to the endpoints. Given this, why is it needed to spread the usage of a modified protocol stack as outlined in this paper?

If our TCP/IP solutions are to be used on wireles networks, we should be able to communicate with any host connected to the Internet without having to know whether the remote host is attached to a low-bandwidth, high-delay link with high BER. This requires changes to existing protocols in end systems.

The changes in link layers need only be done on links where it might be desirable to pass damaged data to the application. In practice, this is usually the access network closest to the endpoints, and changes can therefore be applied where needed without introducing complications to the rest of the Internet as long as they are transparent.

Changes in the network layer can be introduced so that not even the receiving endpoint has to be aware of those changes. By passing the information about insensitive data in a way compatible with IP routers along an arbitrary path and at the destination node, the information can be picked up by the link layers which find it useable.

The changes in the transport layer should not cause any problems for intermediate nodes that do not bother about transport layer headers. However, the partial checksum of UDP Lite cannot be used without support at the destination host. Also, intermediate firewalls using transport layer information as input for Decisions must be configured to be aware of a new transport protocol.

Withthe cost of introducing a new transport protocol in the Internet protocol stack (with all that this means), the benefits would be:

- A more flexible network service for applications that can tolerate damaged data.
- The possibility of using IP for real-time data services [4] in environments where it today is impossible or too costly to get satisfying service quality using IP.

[4] This possibility also gives several positive side effects.

5 Conclusions

In this paper, we have outlined requirements on the Internet protocol stack to make it usable for real-time services in certain wireless environments. We have also presented a few solutions that meet these requirements. By introducing the UDP Lite protocol and a new IP option in the Internet protocol stack, it is possible to use IP as the basis for communication in close to all wireless networks. This would give better connectivity and more cost-efficient solutions compared to using proxy solutions or specialized protocol stacks in wireless environments. Modifications in the network and link layers are also needed, but these can be introduced as needed without disturbing the rest of the Internet.

6 Current and Future Work

We are working on a real implementation of the solutions outlined in this paper in an existing protocol stack. There is also ongoing work to understand the differences in using these mechanisms in various wireless environments. The security aspects need further investigation. We also plan to look into compensating for variations in delay, throughput and BER on wireless IP networks.

References

1. Stephen Casner and Van Jacobson. Compressing IP/UDP/RTP Headers for Low-Speed Serial Links. Request for Comments RFC 2508, Internet Engineering Task Force, February 1999.
2. Stephen Deering and Robert Hinden. Internet Protocol, Version 6 (IPv6). Request for Comments RFC 2460, Internet Engineering Task Force, December 1998.
3. Mikael Degermark, Hans Hannu, Lars-Erik Jonsson, and Krister Svanbro. CRTP over cellular radio links. Internet-Draft (work in progress) draft-degermark-crtp-cellular-01.txt, Luleå University of Technology, December 1999.
4. Mikael Degermark, Björn Nordgren, and Stephen Pink. IP Header Compression. Request for Comments RFC 2507, Luleå University of Technology, February 1999.
5. Carsten Bormann (Ed.). Robust header compression (ROHC). Internet-Draft (work in progress) draft-ietf-rohc-rtp-02.txt, IETF, September 2000.
6. Van Jacobson. Compressing TCP/IP headers for low-speed serial links. Request For Comments RFC 1144, LBL, February 1990.
7. Lars-Erik Jonsson, Mikael Degermark, Hans Hannu, and Krister Svanbro. Robust checksum-based header compression(ROCCO). Internet-Draft (work in progress) draft-ietf-rohc-rtp-rocco-01.txt, Ericsson Research, June 2000.
8. Stephen Kent and Randal Atkinson. Security architecture for the Internet Protocol. Request For Comments RFC 2401, Internet Engineering Task Force, November 1998.
9. Lars-Åke Larzon, Mikael Degermark, and Stephen Pink. Efficient use of wireless bandwidth for multimedia applications. In *Proceedings of the IEEE Mobile Multimedia Communications (MOMUC) Workshop.* IEEE, November 1999.
10. Lars-Åke Larzon, Mikael Degermark, and Stephen Pink. The UDP Lite Protocol. Internet-Draft (work in progress) draft-larzon-udplite-03.txt, Luleå University of Technology, July 2000.

11. Khiem Le. Adaptive header compression (ACE) for real-time multimedia. Internet-Draft (work in progress) draft-ietf-rohc-rtp-ace-00.txt, IETF, May 2000.
12. Jon Postel. Internet Protocol. Request For Comments RFC 791, DARPA, September 1981.
13. The WAP forum homepage, URL: http://www.wapforum.org.

Multicast Routing by Multiple Tree Routes[⋆]

Koohyun Park[1], Yong-Sik Shin[2], and Hyun-Chan Lee[1]

[1] Hong-Ik University, Sangsu-dong 72-1, Mapo-gu,
Seoul, Republic of Korea (121-791)
{khpark, hclee}@wow.hongik.ac.kr
[2] SK Telecom R & D Center, Sunae-dong 9-1, Bundang-gu,
Sungnam-si, Kyunggi-do, Republic of Korea

Abstract. A solution to reduce the cost of multicast paths is presented, where the traffic is split over multiple paths to obtain an overall lower cost. Optimal conditions for this split operation are provided and two different algorithms are presented. The methods were shown to work on problems with many different types of simultaneous multicast traffic. Various experiments were conducted and the results showed that the new multicasting was fairly effective on the end-to-end quality of services.

1 Introduction

Internet Protocol (IP) multicast traffic has increased rapidly with the epochal growth of Internet services. Examples of the wide variety of applications include, Internet movies, Internet broadcasting, video conferences, web-contents contributions, distributive data services, bulletin-board services etc. Since these multicast services include moving-image or video, a broad bandwidth is required and a large amount of multicast traffic should be transmitted. The existing multicast routing methods for the services are based on flooding, minimum spanning trees, source-based routing, or shared trees [7,12].

IP multicasting methods based on the Steiner tree were developed because of their potential efficiency for use in network resources [11,12,13,16,17,20]. Recently Steiner tree problems with end-to-end delay constraints have been studied. Rouskas & Baldine[17] considered the bounded differences in the end-to-end delay. Leung & Yang[13] considered the individual delay-bound for each end-to-end. Parsa et al.[16] considered two measures of delay and cost in the Steiner tree conundrum. They proposed a heuristic method to minimize the total link cost on the tree subject to the end-to-end delay constraints.

The existing IP multicasting methods or Steiner tree based methods find a single tree route in a network with fixed link costs, which are estimated at a certain time. However, the link cost is usually a function of the level of traffic and hence it changes during the transmitting traffic. In most cases, the function is nonlinear with regard to the level of traffic. Hence, the level of traffic should be a major consideration in IP multicasting.

[⋆] This research was supported by Brain Korea 21 grant.

In this study IP multicasting was regarded as a network flow problem, and not as a tree finding problem. The level of traffic demand was considered in our multicast routing problem. Multiple tree routes were used for multicast traffic in order to use efficiently the network resources. The proposed method adds extra tree routes until it satisfies the target conditions. It splits the multicast traffic demand into the chosen tree routes. A mathematical model was developed and the conditions for traffic splitting were optimized. The method is also suitable for problems with many types of different simultaneous multicast traffic. Various experiments were carried and the results suggest show that the new multicasting is fairly effective on the end-to-end quality of services.

The next section introduces the definition of the problem for multicast routing. A mathematical model and its optimality conditions are then developed. In Section 3, the multicasting method with multiple tree routes is proposed. Various experiments are conducted in Section 4 with the final conclusions reported in Section 5.

2 Problem Definition and Optimization Model

2.1 Problem Definition

The following four assumptions were necessary to define the problem of multicasting by multiple tree routes:

Assumtion 1 Multicast traffic is transmitted along a tree that includes a source node and destination nodes. That is, traffic commences from a source node, is copied at a branch node, then transmitted to each destination node.
Assumtion 2 Multicast traffic can be split into multiple trees. However, the traffic in a tree is identical at any link of the tree.
Assumtion 3 Two or more sets of multicast traffic can be simultaneously routed.
Assumtion 4 Cost is incurred only at the links.

Assumption 1 implies that the multicasting problem is a tree based routing. However, Assumption 2 distinguishes the problem from the Steiner tree problem. In the network of Fig. 1, 10 units of multicast traffic are routed from source node 1 to the destination node group {8,9,10}. Let any link cost be incurred as $f(x) = x^2$ when the level of traffic is x in the link. The total link cost incurred in tree (a) of Fig. 1 is then 500 units. In addition, each end-to-end cost is 300 units. However, if two tree routes are used as in (b) of the figure and the traffic number of 10 is split into two 5's at each tree, the total link cost is reduced to 150 units and each end-to-end cost is thus 75 units.

Assumption 2 is meaningless if the link cost is linear in the level of traffic. In this case, the problem becomes a Steiner tree problem in the network with slope of the linear function as the link cost. For the case of a concave link cost, the problem can be transformed into a Steiner tree problem [4]. If the link cost is interpreted as a transfer delay, by queuing theory, most of the link costs incurred in telecommunication networks are convex [10,18]. Convexity is

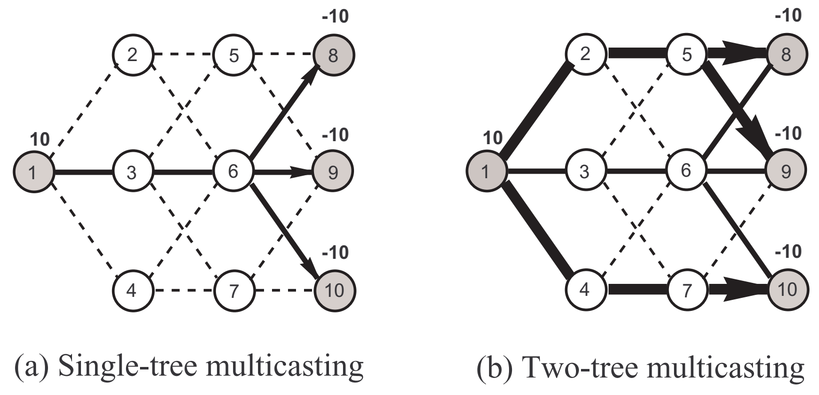


Fig. 1. Multicast routing by a single tree or two trees.

an important condition in this problem. The traffic demand can be optimally split into multiple tree routes with the condition shown in Theorem 2.

By Assumption 3, the problem is a multi-commodity multicasting problem. There are multiple sources and corresponding destination node groups to the sources. In Fig. 2, there are two commodities. One set of multicast traffic is from source node 1 to the destination node group {8,9,10}, denoted briefly as (1,{8,9,10}). The other set of multicast traffic is (11,{2,4}). Let the demand of each multicast be 10 units. At any link, let the cost function be $f(x) = x^2$. The figure compares single-tree multicasting with two-tree multicasting. The total link cost of single-tree multicasting is 1000 units and that of each end-to-end cost is 300 units (see (a)). In (b) of Fig. 1, the traffic is split into two 5's at each tree. Subsequently, the total link cost is reduced to 725 units and each end-to-end cost is one of three values: 75, 150 or 225 units. For example, the 1-to-8 cost is 75 units if the traffic flows along the path 1-3-6-8, and it is 150 units along the path 1-2-5-8. Hence the maximum 1-to-8 cost is 150 units.

In Assumption 4, cost incurring was limited only at the links for simplifying the problem. The node cost can be considered and related comments will be provided for the given occasion.

2.2 Optimization Model

Multicast routing by multiple tree routes is developed by the column generation approach [1]. It is the column generation for adding a new tree route for each (source, group) multicast traffic. Hence an optimization model is updated whenever an additional column is generated. In describing the model, it is necessary to define the following notations.

$G = (V, E)$: Network, V is a set of nodes and E is a set of links. $|E| = m$
d^r: the traffic demand of the rth (source, group) multicast, $r = 1, \ldots, q$

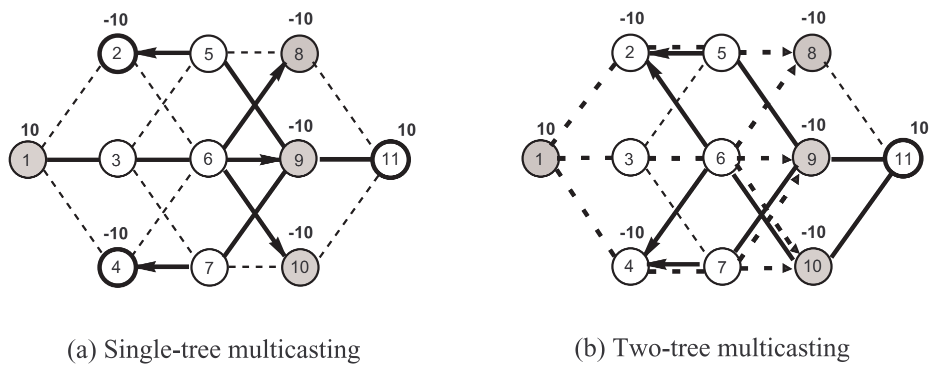


Fig. 2. Multicast routing for two different simultaneous multicast traffic

$S = \{s^1, \ldots, s^q\}$**:** the set of source nodes, s^r is the rth source node,
$D^r(\subseteq V - \{s^r\})$**:** the destination node group of the rth (source, group) multicast, $|D^r| \geq 1$,
$f_i : R \to R$**:** the cost function of the amount of traffic at the ith link, and is assumed to be continuously differentiable,
$P = \cup_{r=1}^{q} P^r$**:** the set of all tree routes,
P^r**:** the set of tree routes for the rth (source, group) multicast, $|P^r| = k,\ r = 1, \ldots, q$,
p_j^r**:** the jth tree route of the rth (source, group) multicast, and is defined by the set of links included in the tree route, $r = 1, \ldots, q$, $j = 1, \ldots, k$,
y_j^r**:** the amount of split traffic into the tree route p_j^r, $r = 1, \ldots, q$, $j = 1, \ldots, k$,
x_i**:** the amount of total traffic passing through the ith link, $i = 1, \ldots, m$.

Suppose that k tree routes are selected for each (source, group) multicast. The optimization model $A(P)$ is as follows:

$$\text{Minimize} \quad \sum_{i=1}^{m} f_i(x_i) \tag{1}$$
$$\text{Subject to} \quad \sum_{j=1}^{k} y_j^r = d^r, r = 1, \ldots, q, \tag{2}$$
$$\sum_{r=1}^{q} \sum_{j:i \in p_j^r} y_j^r = x_i, i = 1, \ldots, m, \tag{3}$$
$$x_i \geq 0, y_j^r \geq 0, i = 1, \ldots, m, j = 1, \ldots, k, r = 1, \ldots, q. \tag{4}$$

Here P is the set of total qk tree routes. The model $A(P)$ is an optimization problem upon the stage that k tree routes are added to P for each (source, group) multicast. Its optimality conditions will be developed in the next subsection with the intention of using them in an algorithm. The objective function, formula (1), denotes the total link cost incurred in the network, and equations (2) are the constraints to satisfy the demand for each (source, group) set of multicast traffic. In equation (3), the traffic passing through a link is accumulated.

Considering the node cost, $\sum_{l=1}^{n} \psi_l(w_l)$ is added to the objective function and new constraints $\sum_{r=1}^{q} \sum_{j:l \in p_j^r} y_j^r = w_l (l = 1, \ldots, n)$ are appended. Here $w_l (l = 1, \ldots, n)$ is the level of traffic passing through the corresponding node l, $\psi_l(w_l)$ is the node cost incurred at the lth node. In the model with node cost, p_j^r is defined by the set of links and nodes included in the tree route.

2.3 Optimality Conditions

In order to develop the optimality conditions, it is necessary to define the gradient of a tree route at a given traffic flow. Assume that there is a given link flow x and a split tree flow y, which are feasible solutions to the model $A(P)$. Here $x = (x_1, \ldots, x_m)$, $y = (y^1, \ldots, y^q)$ and $y^r = (y_1^r, \ldots, y_k^r)$.

Definition 1. *The change in the total link cost as the traffic in a tree route p_j^r increases by one unit, $\sum_{i \in p_j^r} f_i'(x_i)$ is defined by* ***the gradient of the tree route*** *p_j^r at the given traffic flow (x, y).*

Theorem 1. *Suppose that (x, y) is an optimal solution of the model $A(P)$ and for each i, f_i is strictly increasing. Then for each r there exists $\pi^r > 0$ such that $\sum_{i \in p_j^r} f_i'(x_i) = \pi^r$ for the tree route $p_j^r \in P^r$ with $y_j^r > 0$, and $\sum_{i \in p_j^r} f_i'(x_i) \geq \pi^r$ for the tree route $p_j^r \in P^r$ with $y_j^r = 0$.*

Proof. By the Karush-Kuhn-Tucker[3] conditions for the model $A(P)$, there exist $\pi^r (r = 1, \ldots, q)$ and $v_i (i = 1, \ldots, m)$ satisfying the following:

$$f_i'(x_i) + v_i \geq 0, \ i = 1, \ldots, m \tag{5}$$

$$-\pi^r - \sum_{i \in p_j^r} v_i \geq 0, \ j = 1, \ldots, k, r = 1, \ldots, q, \tag{6}$$

$$x_i(f_i'(x_i) + v_i) = 0, \ i = 1, \ldots, m \tag{7}$$

$$y_j^r(-\pi^r - \sum_{i \in p_j^r} v_i) = 0, \ j = 1, \ldots, k, r = 1, \ldots, q. \tag{8}$$

Here π^r is the multiplier of the rth equation of (2), and v_i is the multiplier of the ith equation of (3). For each tree route p_j^r with $y_j^r > 0$, the following equation has the from (8):

$$-\sum_{i \in p_j^r} v_i = \pi^r, \ j = 1, \ldots, k, r = 1, \ldots, q. \tag{9}$$

Since every link i in p_j^r with $y_j^r > 0$ has positive flow, $x_i > 0$ for $i \in p_j^r$. From (7), we have $f_i'(x_i) = -v_i$ for the link. Substituting this in (9), for p_j^r with $y_j^r > 0$ we have

$$\sum_{i \in p_j^r} f_i'(x_i) = \pi^r. \tag{10}$$

Since $f_i(\cdot)$ is strictly increasing in the range $x_i > 0$, $f_i'(x_i) > 0$. Therefore, $\pi^r > 0$.

For each p_j^r, (5) and (6) give

$$\sum_{i \in p_j^r} f_i'(x_i) \geq -\sum_{i \in p_j^r} v_i \geq \pi^r. \tag{11}$$

Therefore, this is clearly satisfied for each p_j^r with $y_j^r = 0$. □

For an optimal solution (x, y) of the model $A(P)$, denote P_+^r as the set of tree routes $p_j^r \in P^r$ with $y_j^r > 0$ and denote P_0^r as the set of tree routes $p_j^r \in P^r$ with $y_j^r = 0$. Thus the theorem can be rewritten as follows: For each rth (source, group) multicast, there exists $\pi^r > 0$ such that the gradient of any tree route in P_+^r is π^r and that of any tree route in $P_0^r \geq \pi^r$.

Considering the node cost, the gradient of a tree route p_j^r is defined by $\sum_{i \in p_j^r} f_i'(x_i) + \sum_{l \in p_j^r} \psi_l'(w_l)$. Therefore, by the same rationale the theorem is revised as follows: For each r there exists $\pi^r > 0$ such that $\sum_{i \in p_j^r} f_i'(x_i) + \sum_{l \in p_j^r} \psi_l'(w_l) = \pi^r$ for the tree route $p_j^r \in P^r$ with $y_j^r > 0$ and $\sum_{i \in p_j^r} f_i'(x_i) + \sum_{l \in p_j^r} \psi_l'(w_l) \geq \pi^r$ for the tree route $p_j^r \in P^r$ with $y_j^r = 0$.

Theorem 2. *(x, y) is a feasible solution of the model $A(P)$, and for each i, f_i is strictly increasing and convex. Suppose that for each rth (source, group) multicast, there exists $\pi^r > 0$ such that the gradient of any tree route in P_+^r is π^r and that of any tree route in P_0^r is greater than or equal to π^r. Then (x, y) is a global optimal solution to the model $A(P)$.*

Proof. From the hypothesis of the theorem, we have

$$\sum_{i \in p_j^r} f_i'(x_i) = \pi^r, \text{for each } p_j^r \text{ with } y_j^r > 0 \tag{12}$$

$$\sum_{i \in p_j^r} f_i'(x_i) \geq \pi^r, \text{for each } p_j^r \text{ with } y_j^r = 0 \tag{13}$$

First, we need to demonstrate that these two conditions imply the Karush-Kuhn-Tucker conditions (5)–(8). For each link i with $x_i > 0$, let $v_i = -f_i'(x_i)$. And for the link i with $x_i = 0$, choose v_i such that $v_i \geq -f_i'(x_i)$ and $\sum_{i \in p_j^r} v_i \leq -\pi^r$. This is possible since $-\sum_{i \in p_j^r} f_i'(x_i) \leq -\pi^r$ from (13). Then conditions (5)–(7) are satisfied. Now for each tree route p_j^r with $y_j^r > 0$, we have $-\sum_{i \in p_j^r} v_i = \sum_{i \in p_j^r} f_i'(x_i) = \pi^r$ from (12). This satisfies (8).

Secondly, the conditions (5)–(8) can be shown to be the optimality sufficient conditions to the model $A(P)$. Note that the objective function of the model is convex and all functions of constraints are linear. These satisfy the Karush-Kuhn-Tucker sufficient conditions in Theorem 4.2.16 of [3]. Therefore, the feasible solution (x, y) satisfying the hypothesis in our theorem is a global solution of the model. □

Theorem 3. *Suppose that (x, y) is an optimal solution of the model $A(P)$ and for each i, f_i is strictly increasing. Select one tree route p_j^r from each P^r. Then there exists a positive $\pi^1, \ldots, \pi^q$ and $\hat{\pi}$ such that*
(i) if $p_j^r \in P_+^r$ for all r, $\sum_{r=1}^q \pi^r = \hat{\pi}$,
(ii) if $p_j^r \in P_0^r$ for all r, $\sum_{r=1}^q \pi^r \geq \hat{\pi}$.

Proof. For each rth (source, group) multicast, the gradient of any tree route $p_j^r \in P_+^r$ is identically equal to $\pi^r = \sum_{i \in p_j^r} f_i'(x_i)$. This is satisfied independently of j by Theorem 1. Therefore, the total gradients of the selected tree routes with a positive flow should be equal to $\sum_{r=1}^q \pi^r = \hat{\pi}$. Now assume that there exists one tree route with zero flow among the set of selected tree routes. Let it be in the rth (source, group) multicast. Therefore, it is in P_0^r. Thus the gradient of the tree route $\geq \pi^r$. This provides the second case (ii) of the theorem. □

3 Algorithms

In this section two algorithms were developed for multicasting by multiple tree routes. The algorithms are based on the column generation approach [1]. Initially they begin without any tree routes. However, a tree route for each (source, group) multicast is added during the iteration. For real applications, the number of tree routes was limited. Let L be the maximum number of tree routes for each (source, group) multicast. The first algorithm is performed until either the conditions in Theorem 3 are satisfied or the number of tree routes is reached at the maximum value L. The second algorithm is a revision of the first. In this algorithm, the first algorithm is applied with the limit number $L' \geq L$. Therefore, L tree routes are selected among L' tree routes in the order of their traffic magnitude for each (source, group) multicast. Finally the traffic demand is split with the L tree routes.

In the algorithms, t is the current iteration number, which denotes the number of tree routes added for each (source, group) multicast. P_t^r denotes the set of the tree routes for the rth (source, group) multicast in iteration t. An $m \times t$ link-tree incidence matrix H_t^r is defined of which the (i, j)th element is 1 if the ith link is included in $p_j^r \in P_t^r$ otherwise it is 0. Some notations and relations are given for algorithms:

$x^t = \sum_{r=1}^q H_t^r y^r$, where the ith element of x^t is the total level of traffic passing through the link.
$dist(i)$ is the distance of link.
$\hat{\pi}_t = \sum_{r=1}^q \pi_t^r$ where $\pi_t^r = \sum_{i \in p_j^r} f_i'(x_i^t)$ and p_j^r is any tree route in P_t^r.

Algorithm I

(Step 1) Choose L.
(Step 2) Let $dist(i) = f_i'(0)$ for each link i. Find a tree route p_1^r for each $r = 1, \ldots, q$. Set $P_1^r = \{p_1^r\}$, and let $y_1^r = d^r$. Determine H_1^r.
(Step 3) Compute x^1, $\pi_1^r (r = 1, \ldots, q)$ and $\hat{\pi}_1$. Set $t = 1$.

(Step 4) Let $dist(i) = f_i'(x_i^t)$ for each link i. Find a candidate tree p_{t+1}^r for each $r = 1, \ldots, q$.
(Step 5) If the sum of the gradients of the q candidate trees is $\geq \hat{\pi}_t$, stop and printout the current solution (x, y). Otherwise, update $P_{t+1}^r = P_t^r \cup \{p_{t+1}^r\}$ and H_{t+1}^r for each $r = 1, \ldots, q$.
(Step 6) Put $P = \cup_{r=1}^q P_{t+1}^r$. Solve the model $A(P)$ and let its optimal solutions be (x^{t+1}, y^{t+1}). Compute $\pi_{t+1}^r (r = 1, \ldots, q)$ and $\hat{\pi}_{t+1}$ at the optimal solution.
(Step 7) Stop if $t + 1 \geq L$. Otherwise, put $t = t + 1$ and go to step 4.

In order to apply the algorithm to the multicasting problem in this study, two parts of it should be implemented. One part is to find a candidate tree route for each (source, group) set of multicast traffic in both step 2 and step 4. A Steiner tree problem is required to find the best candidate tree. However as is well known, the Steiner tree problem is NP-hard[5]. Therefore, any heuristic method may be used for the algorithm. Wong's method [22] was adopted in this study since it can achieve very good near-optimal solutions and provide the lower bounds of the true optimal value.

The second part to be implemented is to solve the model $A(P)$ in step 6. In this case, the iterative quadratic programming method [6,14] was applied. For each iteration, a quadratic program is obtained by approximating the objective function as a quadratic function. CPLEX [8] was adopted for solving the approximated problems. Global convergence is guaranteed if the link cost functions satisfy the hypothesis of Theorem 2 and are strictly convex [6].

Algorithm II is proposed for improving the first algorithm.

Algorithm II

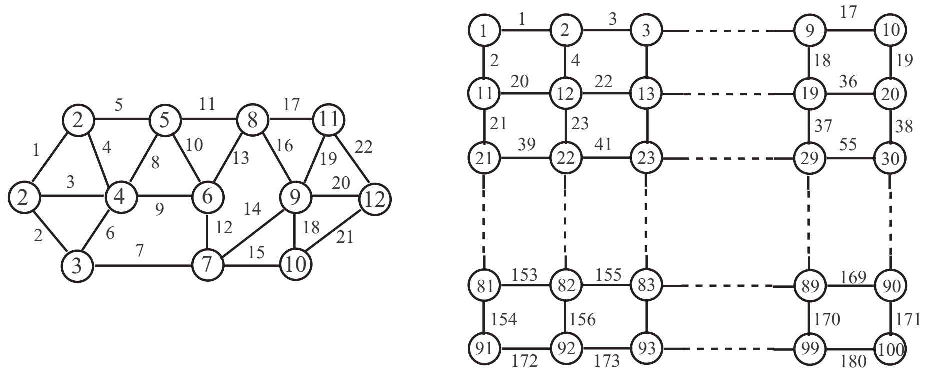
(Step 1) Apply Algorithm I with the value $L' \geq L$.
(Step 2) Obtaining P^r after executing Algorithm 1, sort the tree routs in the order of traffic magnitude for each $r = 1, \ldots, q$.
(Step 3) Select orderly tree routes up to L in each P^r and let it be $\tilde{P^r}$. Set $\tilde{P} = \cup_{r=1}^q \tilde{P^r}$.
(Step 4) Solve the model $A(P)$.

Even though more tree routes are selected than L(up to L'), L tree routes are used for each (source, group) multicast.

4 Experimental Results

Experiments were conducted to test whether the algorithms could be applied to various multicasting problems. The two networks shown in Fig. 3 were considered. Network I has twelve nodes and twenty-two links. Network II is a very large one with 100 nodes and 180 links.

Three types of link cost functions are considered in Table 1. In particular, the fractional function is often used to represent the transfer delay in telecommunication networks [10,18]. In the table, $rand(0, 1)$ is the random number uniformly



(a) Network I (b) Network II

Fig. 3. Multicast routing for two different simultaneous multicast traffic

generated in [0,1]. It is used in selecting the coefficients of the link cost functions. All of the functions in the table are strictly increasing and strictly convex. Therefore the algorithms converge to a solution.

Table 1. Cost functions at each link

Function type	Function	Coefficients
Quadratic	$a_i x_i^2 + b_i x_i, (x_i \geq 0)$	$a_i = 1 + rand(0,1)$
		$b_i = 1 + rand(0,1)$
Fractional	$x_i/(c_i - x_i), (0 \leq x_i < c_i)$	$c_i = 10 + rand(0,1)$
Exponential	$\exp(x_i/c_i) - 1, (x_i \geq 0)$	$c_i = 1 + rand(0,1)$

Table 2 shows experiments with a single (source, group) multicast, whose experiment numbers begin with the letter 'S'. Table 3 shows experiments with multiple (source, group) multicast, whose experiment numbers start with the letter 'M'. The 'Numbers of destination nodes' of the third column in Table 3 do not mean the serial node numbers but the number (quantities) of nodes. For example, in experiment M2, there are three source nodes. Corresponding to the sources, there are two nodes, three nodes and three nodes as destinations, respectively.

Table 4 shows the results of Algorithm I to the experiments in Tables 2 and 3. In the application the stopping rule of step 7 in the algorithm was abandoned. Therefore, the algorithm terminates only when it satisfies the conditions of Theorem 3 in step 5. These results would be same as those for the cases with a sufficiently large number L.

Table 2. Experiments with single (source,group) multicast

Experiment number	Source node	Destination nodes	Traffic demand	Network	Link cost functions
S1	1	12	8	I	Quadratic
S2	1	3,7,12	10	I	Quadratic
S3	2	12	8	I	Fractional
S4	5	1,7,12	8	I	Fractional
S5	2	99	15	II	Exponential
S6	5	21,30,81,90,9	15	II	Exponential

Table 3. Experiments with multiple (source,group) multicast

Experiment number	Number of simultaneous multicast	Number of destination nodes	Multicast traffic demands	Network	Link cost functions
M1	2	1,1	7,5	I	Quadratic
M2	3	2,3,3	10,5,5	I	Quadratic
M3	2	1,1	3,5	I	Fractional
M4	4	1,2,3,4	2,2,2,3	I	Fractional
M5	3	2,2,4	5,7,10	II	Exponential
M6	5	1,2,3,4,5	3,5,5,10,10	II	Exponential

Table 4. Experimental results with sufficient large L

Experiment number	Number of tree routes	The total link cost
S1	11	150.141
S2	8	214.953
S3	9	4.539
S4	6	6.626
S5	75	455.285
S6	17	2020.158
M1	8,5	139,441
M2	3,3,2	353.280
M3	3,5	4.937
M4	4,4,1,3	5.690
M5	8,11,12	1013.547
M6	12,7,8,14,11	6006.344

If multiple tree routes are used for a (source, group) multicast, then there are multiple paths for each source-to-destination. Therefore, the end-to-end paths have different end-to-end costs. In this study, the end-to-end QoS was defined as the maximum end-to-end cost. Table 5 shows the changes in the end-to-end QoS's accordingly as $L = 1, 2, 3$. As shown in the tables only two or three tree routes give successful results in end-to-end QoS.

Table 5. Changes in end-to-end QoS according to $L = 1, 2, 3$ for experiments S4 and M5

Experiment number	End-to-end	End-to-end QoS $L=1$	$L=2$	$L=3$
S4	$5 \to 1$	5.90	1.35	1.30
	$5 \to 7$	12.50	2.50	2.40
	$5 \to 12$	18.00	3.50	3.30
M5	$2 \to 90$	379	160	158
	$2 \to 92$	211	88	71
	$9 \to 21$	2300	971	297
	$9 \to 50$	505	74	51
	$95 \to 11$	2200	711	227
	$95 \to 20$	2500	1159	315
	$95 \to 55$	2100	289	111

In Table 6, Algorithm I is compared with Algorithm II for experiments S2, S4, M3 and M5. It shows changes in the total link cost according to the number of tree routes L. The number L' in Algorithm II is used as the number of tree routes from the results in Table 4. The results of the two algorithms do not differ greatly, and Algorithm II is usually superior to Algorithm I.

Table 6. Changes in the total link cost according to the number of tree routes L for experiments S2, S4, M3 and M5

Experiment number	Algorithm	Total link cost $L=1$	$L=2$	$L=3$	$L=4$	$L=5$
S2	Algorithm I	700	323	279	241	228
	Algorithm II	600	300	272	240	219
S4	Algorithm I		8.151	7.616	7.467	7.200
	Algorithm II		20.000	7.628	7.233	7.164
M3	Algorithm I	5.996	5.071	5.004	4.947	4.937
	Algorithm II	5.996	5.261	4.941	4.938	4.937
M5	Algorithm I		3435	1640	1306	1158
	Algorithm II		2958	1327	1174	1109

Table 7 shows the effect of the multiple tree routes on the total link cost. $F(1)$ denotes the total link cost incurred with a single tree route, and $F(L)$ denotes the total link cost incurred with an L tree route. Algorithm II was applied for multicasting with multiple tree routes. As the results are similar to the those in the end-to-end QoS, the $F(L)/F(1)$ ratio reduces sharply though L is two or three. Subsequently, the ratio is steady even when L increases. For experiments S4 and M5, only two tree routes for each (source, group) multicast brings a 90% overall decrease in the total link cost.

Table 7. Reduction effects of multiple tree routes L for experiments S2, S4, M3 and M5

Experiment	$F(L)/F(1)$				
number	$L = 1$	$L = 2$	$L = 3$	$L = 4$	$L = 5$
S2	1.00	0.50	0.45	0.40	0.37
S4	1.00	0.08	0.08	0.07	0.07
M3	1.00	0.85	0.84	0.83	0.83
M5	1.00	0.05	0.02	0.02	0.02

5 Conclusions

Recently Internet service demand has increased rapidly. As a consequence, IP multicast traffic has likewise increased. Multicast traffic requires a broad bandwidth and a strict end-to-end QoS. If multicasting is accomplished by single tree routing as is currently with existing multicasting methods, depending on the level of multicast traffic, the end-to-end QoS can deteriorate seriously. Therefore, a multicasting method was proposed using multiple tree routes. In this method the amount of traffic is considered. That is, in this study, IP multicasting was regarded as a network flow problem, not as a tree finding problem. The proposed method adds tree routes until it satisfies the target conditions, and it splits optimally the multicast traffic demand into the chosen tree routes. The method is also intended for problems with many different types of simultaneous multicast traffic.

Various experiments were conducted and the results show that the new multicasting method is fairly effective on the end-to-end quality of services as well as the total link cost. Two or three tree routes give successful results. In some experiments, only one more tree route (i.e., two tree routes) results in over a 90% improvement in the end-to-end QoS and the total link cost.

References

1. Ahuja, R.K., Magnanti, T.L., Orlin, J.B.: Network Flows: Theory, Algorithms, and Applications, Prentice-Hall (1993)
2. Ammar, M.K., Cheung, S.Y., Scoglio, C.M.: Routing multipoint connections using virtual paths in an ATM networks, IEEE INFOCOM **'93** (1993) 98–105
3. Bazaraa, M.S., Sherali, H.D., Shetty, C.M.: Nonlinear Programming: Theory and Algorithms, Wiley (1993)
4. Erickson, R. E., Monma, C. L., Veinott, A. F.: Send-and-split method for minimum-concave-cost network flows, Mathematics of Operations Research **12** (1987)
5. Garey, M.R, Johnson, D.S.: Computers and Intractability: A Guide to the Theory of NP-Completeness, W.H. Freedman and Company (1979)
6. Han, S.P.: A gobally convergent method for nonlinear programming, Journal of Optimization Theory and Applications **22** (1977) 297–309
7. Huitema, C.: Routing in the Internet, Prentice Hall (1995)
8. Using the CPLEX Callable Library, ILOG, Inc. (1997)
9. Kim, S.B.: An optimal VP-based multicast routing in ATM networks, IEEE INFOCOM **'96** (1996) 1302–1309
10. Kleinrock, L.: Queueing Systems(Volume II): Computer Applications , Wiley (1976)
11. Kompella, V.P., Pasquale, J.C., Polyzos, G.C.: Multicast routing for multimedia communication, IEEE/ACM Transaction on Networking **1** (1993) 286–292
12. Kosiur, D.: IP Multicasting, Wiley, (1998)
13. Leung, Y.-W., Yang, B.-T: Lower bound for multimedia multicast routing, IEE Proc.-Commm. **145** (1998) 87–90
14. Mangasarian, O.L.: Nonlinear Programming Algorithm, Lecture Notes, Department of Computer Science, University of Wisconsin-Madison (1986)
15. Park, K., Shin, Y.-S.: Iterative bundle-based decomposition for large-scale nonseparable convex optimization, European Journal of Operational Research **111** (1998) 598–616
16. Parsa, M., Zhu, Q.: Garcia-Luna-Aceves, J.J., An iterative algorithm for delay-constrained minimum-cost multicasting, IEEE/ACM Transaction on Networking **6** (1998)
17. Rouskas, G. N., Baldine, I.: Multicast routing with end-to-end delay and delay variation constraints, IEEE INFOCOM **'96** (1996) 353–360
18. Schwartz, M.: Telecommunication Networks: Protocols, Modeling and Analysis, Addison Wesley (1988)
19. Takahashi, H., Matsuyama, A.: An approximation solution for the Steiner problem in graphs, Math. Japonica **24** (1980) 573–577
20. Waxman, B.M.: Routing of multipoint connections, IEEE Journal on Selected Areas in Communications **6** (1988) 1617–1622
21. Winter, P.: Steiner problem in networks: A survey, Networks **17** (1987) 129–167
22. Wong, R.T.: A dual ascent approach for Steiner tree problem on a directed graph, Mathematical Programming **28** (1984) 271–284

Optimal Design of Optical Ring Networks with Differentiated Reliability (DiR)*

Andrea Fumagalli and Marco Tacca

Center for Advanced Telecommunications Systems and Services (CATSS)
University of Texas at Dallas
Erik Jonsson School of Engineering and Computer Science
{andreaf,mtacca}@utdallas.edu

Abstract. Current optical networks typically offer two degrees of service reliability: full protection in presence of a single fault in the network, and no protection at all. This situation reflects the historical duality that has its roots in the once divided telephone and data environment. The circuit oriented service required *protection,* i.e., provisioning of readily available spare resources to replace working resources in case of a fault. The datagram oriented service relied upon *restoration,* i.e., dynamic search for and reallocation of affected resources via such actions as routing table updates.
The current development trend, however, is gradually driving the design of networks towards a unified solution that will jointly support traditional voice and data services as well as a variety of novel multimedia applications. The growing importance of concepts, such Quality of Service (QoS) and Differentiated Services that provide varying levels of service performance in the same network evidences this trend.
Consistently with this pattern, the novel concept of *Differentiated Reliability (DiR)* is formally introduced in the paper and applied to provide multiple reliability degrees (classes) in the same network layer using a common protection mechanism, i.e., path switching. According to the DiR concept, each connection in the layer under consideration is guaranteed a minimum reliability degree, defined as the Maximum Failure Probability allowed for that connection. The reliability degree chosen for a given connection is thus determined by the application requirements, and not by the actual network topology, design constraints, robustness of the network components, and span of the connection.
An efficient algorithm is proposed to design the Wavelength Division Multiplexing (WDM) layer of a DiR ring.

1 Introduction

Current statistics reveal that businesses increasingly rely on communications networks. E-commerce and e-business are heavily dependent upon the availabil-

* The work was partially supported by the Texas Higher Education Coordinating Board under contract # 009741-0139-1999, TxTEC, and NSF under contract # ANI-0082085.

ity of communication resources. Thus, reliable communication networks are a must from the end user's point of view.

In a multi-layer network architecture, protection switching techniques may be implemented in more than a single layer. For instance, in the IP (Internet Protocol) over WDM (Wavelength Division Multiplexing) architecture it is possible to implement protection[1] mechanisms at the WDM layer [6], or a set of distinct protection mechanisms via Multi Protocol Label Switching (MPLS) at the IP layer [9], or restoration[2] mechanisms in the conventional IP manner [12]. With these options available, a competitive and cost effective design of the reliable network is a critical goal that may determine the success or failure of the proposed network architecture. In other words, which layer should provide protection and what degree of protection does each layer need? It is well known that distinct protection mechanisms will coexist in the same network, with each mechanism being implemented at a distinct layer [2]. One rule of the thumb is that using the protection mechanism at the lowest possible layer minimizes the response time [3,1].

Little is known, however, about the degree of reliability required at each layer, that is, the level of reliability that each layer should provide to higher layers. Clearly, there is a tradeoff between the degree of reliability that is offered by each layer and the cost of the resources required at that layer. A higher degree of reliability comes at higher cost. Another factor to consider is that the same network (or layer) is designed to support multiple services, and each service might require a distinct degree of reliability. For example, the WDM layer may support both SONET and IP layers, and may provide to each higher layer a different degree of reliability.

Currently, only two degrees of service reliability are commonly considered: 100% protection in presence of a single fault in the network and none at all. These two approaches have their historical origin in the once divided telephone and data environments. The former service, being circuit oriented, requires protection for immediate network restoration upon a fault. The latter service, being datagram oriented, relies upon restoration, i.e., dynamic updates of the network routing tables. The current trend is gradually driving the design of networks towards a unified solution that will support, not only traditional voice and data services, but also a variety of novel multimedia applications. Evidence of this trend over the last decade is the introduction of concepts like Quality of Service (QoS) [10,8] and Differentiated Services [7,4] to provide varying levels of service performance in the same network. Surprisingly, little has been discussed or proposed for network reliability schemes to accommodate this change in the way networks are being designed.

[1] *Protection* indicates a technique based on the provisioning of readily available resources to replace specific working connections in case of a fault.

[2] *Restoration* indicates a technique based on the dynamic search for alternative resources that will allow to continue the connections affected by the fault.

Today's competitive networks can no longer provide just pure circuit switching and datagram services, nor they can limit the options of reliability to only two degrees, either fully protected or unprotected.

In this paper the problem of designing cost effective multi-layer networks that are capable of providing various reliability degrees — as opposed to 0% and 100% only — is addressed. The concept of *Differentiated Reliability (DiR)* is for the first time formally introduced and applied to provide multiple reliability degrees (or classes) at the same layer using a common protection mechanism, i.e., path switching [13,11].

According to the DiR concept, each connection in the layer under consideration is assigned a Maximum Failure Probability (MFP) which is defined as the probability that the connection is unavailable due to the occurrence of a fault in the network. The MFP chosen for a given connection is determined by the application requirements and not by the protection mechanism[3], the network topology, the source-destination distance, or the Mean Time Between Failure (MTBF) of the network components. It is expected that the cost of the connection is inversely proportional to the chosen MFP degree. For example, in a simplified network scenario with only two layers, i.e., IP over WDM, an optical connection between two IP routers shall guarantee the MFP required by the user. Certain connections between routers may be more critical than others, thus, a cost effective solution will rely on some highly reliable optical connections and other less reliable connections whose disruption will not heavily affect the user's IP network.

From a practical point of view, DiR connections can rely upon the protection mechanism preferred by the network designer, or available from technology. Connections are then individually routed and assigned spare resources in such a way that the MFP degree required for each connection is met. Preemption of a less reliable connection to guarantee reliability of a higher reliable connection is allowed when the resulting MFP degree still meets the connection requirement.

In summary, the DiR concept offers the unique combination of the following advantages. The user (or upper layer) determines the desired MFP degree for each connection. Thus, in a multi-layer network, the lower layer may provide the above layers with the desired reliability degree, transparently from the actual network topology, design constraints, device technology and connection span. In addition, DiR allows to dynamically adjust the network configuration to take into account possible improvements of the network component's MTBF that may become available in some portion of the network, without affecting the reliability degree offered to the upper layers. In practical terms, improved MTBF of network components may allow the same network to support additional connections without affecting the guaranteed MFP degree of the already existing connections.

In the rest of the paper the DiR concept is demonstrated using a bidirectional WDM ring in which protection is achieved in the optical layer.

[3] With DiR, the protection mechanism may offer a variety of different MFP degrees

2 Applying the DiR Concept to WDM Rings

First the problem of optimally designing the WDM DiR-based ring is defined, then an efficient algorithm that sub-optimally solves the problem is presented.

2.1 The Problem of DiR Optimal Design

This subsection describes the assumptions made and defines the DiR optimal design problem.

It is assumed that a set of connection demands are given and must be routed across the ring. A demand consists of one or multiple lightpaths[4], that need to be established between two nodes. Each demand is assigned a requested MFP that must be met by the protection mechanism at the optical layer. Only one protection mechanism is available at the optical layer to achieve the demand requested MFP. This requirement appears to be necessary to provide feasible network management as opposed to complex management handling of multiple concurrent recovery actions that take place in the event of a network element failure. The protection mechanism considered in the paper is the $1:1$ path protection applied to lightpaths. With this protection mechanism, a *working lightpath* is assigned a route-disjoint *protection lightpath* that may be alternatively used if the working lightpath fails to work.

First, the set of multiple *reliability classes* is introduced. A reliability class, c, is characterized by a connection maximum failure probability $MFP(c)$, which indicates the maximum acceptable probability that, upon a network element failure, a connection in that class will not survive despite the optical layer protection[5].

Connection demands are assigned to classes according to their required MFP. Traffic demands in higher requirement classes, i.e., lower MFP, are those with the most stringent requirements in terms of recovery speed and need to be restored at the optical layer (e.g., voice traffic), as opposed to traffic demands in lower classes that may also be recovered by means of the (relatively slower) IP restoration mechanisms.

The WDM ring topology is modeled as a graph $G(\mathcal{N}, E)$. Set $\mathcal{N}$ represents the network nodes and link set E represents the WDM ring lines connecting physically adjacent nodes. Each link in E is characterized by two numbers: link cost and link failure probability.

The link cost represents the cost of routing a lightpath on that link. For example, if the link cost is set equal to the link length, it is assumed that the cost of routing a lightpath on that link is proportional to the link length.

The link failure probability is the probability that the considered link is faulted under the condition that a single network line fault has occurred in the ring. (The analysis presented here is based on the assumptions that only single

[4] A lightpath is a path of light between a node pair, whose bandwidth equals the wavelength bandwidth.

[5] The concept of reliability class can be easily generalized to multiple failures.

(line) faults may occur. However, the proposed technique can be extended to handle concurrent faults of varying natures, including node faults.) The link failure probability is estimated on the basis of available failure statistics of the employed optical components. First, the probability of having a single fault in the network is estimated. Once this value is known, every link fault probability is normalized to the probability of having a single fault in the network. We define $P_f(i,j)$ as the failure probability of link (i,j), given the occurrence of one fault in the network. A uniform distribution of faults across the link, would then result in $P_f(i,j) = \frac{1}{|E|} \;\; \forall (i,j) \in E$.

To demonstrate the DiR concept, the $1:1$ path protection scheme is modified as follows. (Notice that provisioned protection resources are unused in absence of network failure.) A lower class working lightpath may be routed on already provisioned protection wavelengths (Fig. 1), thus improving use efficiency of network resources. Consequently, in case of a link failure, a higher class connection may preempt a lower class connection if the latter is using protection resources dedicated to the former. This mechanism affects the connection failure probability of the lower class connections due to its preemption. From a lower class connection viewpoint, preemption is equivalent to a fault that disrupts its working lightpath without the possibility to resort to a protection lightpath. For each connection, we thus define the connection failure probability as the sum of the failure probabilities of its unprotected links plus the probability of "virtual" failure due to preemption.

In Fig. 1, we show an instance of the ring with uniform line failure distribution. The higher class connection (thick line) has $MFP = 0$, i.e., it must be 100% protected against any single line fault. The lower class connection (thin line) reuses protection wavelengths assigned to the higher class connection (lines (D,C) and (C,B)), thus its failure probability is given by the sum $P_f(D,C) + P_f(C,B)$ (fault of the unprotected links of the lower class connection), plus the probability of being preempted by the higher class connection which is $P_f(D,E) + P_f(E,A)$.

The objective of the DiR problem is to determine the routing for and the resources used by each lightpath request in order to minimize the ring total *wavelength-mileage*, or Λ — more generally, the network cost — subject to guaranteeing the requested reliability degree (MFP) of each traffic class c.

The objective function is formally described next under the assumption that wavelength converters are available at all nodes and any (working or protection) lightpath may be assigned a distinct wavelength on each of its links.

$$min \; \Lambda = \sum_{i,j} \left(\sum_{c=1}^{C} \sum_{s,d} (w_{i,j}^{s,d,c} + p_{i,j}^{s,d,c} - r_{i,j}^{s,d,c}) \cdot l_{i,j} \right) \qquad (1)$$

where:

- $w_{i,j}^{s,d,c}$ is the number of (working) wavelengths on line (i,j) assigned to the working lightpaths that belong to the connection demand from s to d of class c

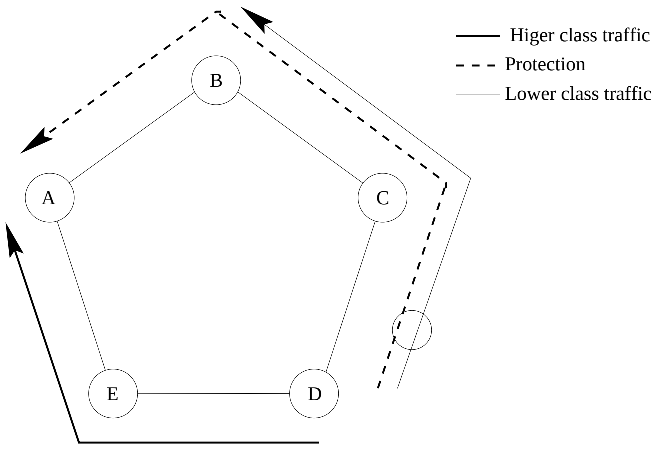


Fig. 1. Protection and preemption mechanism

- $p_{i,j}^{s,d,c}$ is the number of (protection) wavelengths on line (i,j) assigned to the protection lightpaths that belong to the connection demand from s to d of class c
- $r_{i,j}^{s,d,c}$ is the number of protection wavelengths on line (i,j) that are reused by working lightpaths that belong to the connection demand from s to d of class c
- $l_{i,j}$ is the length of line (i,j).

The objective function must be minimized under the constraint that the probability failure of every connection demand in class c does not exceed the requested maximum failure probability $MPF(c)$.

2.2 The Difficult-Reuse-First Algorithm

This section presents an efficient greedy algorithm that may be used to sub-optimally[6] solve the DiR design problem in WDM rings. The name of the algorithm is Difficult-Reuse-First (DRF) to indicate that, prior to being routed, demands are sorted considering how difficult it is for the corresponding working lightpaths to achieve reuse of protection wavelengths. Lightpaths that are more difficult in that regard, are routed first, thus giving them a better chance to achieve a more efficient wavelength reuse.
The DRF algorithm is based on two observations.

1. Due to the topological layout of bidirectional ring, only two disjoint routes exist between any node pair (Fig. 2). Routing is thus a binary problem and each demand that requires protection must employ both routes, one for the working lightpath and the other for the protection lightpath (Fig. 2(b)).

[6] The complexity of the DiR design problem is under investigation.

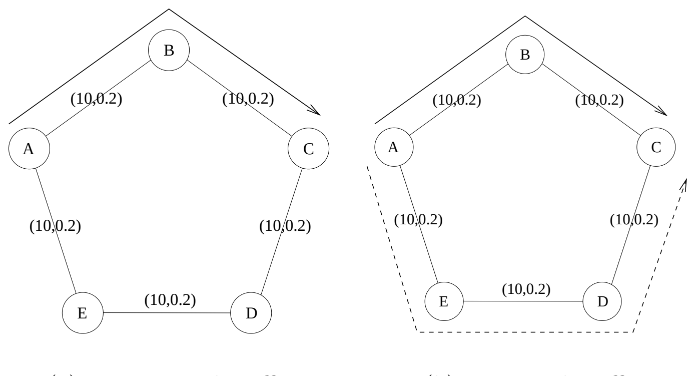


Fig. 2. Ring protection mechanism

2. When the reliability degree requested for the connection cannot be achieved using the already provisioned wavelengths, an additional wavelength must be added to the ring. Under this circumstance, irrespective of the routing (clockwise or counterclockwise) chosen for the working (protection) lightpath, one wavelength must be added to every line of the ring. The cost (working and protection) for setting the connection thus equals the cost of adding a wavelength along the whole ring perimeter.

The following definitions are used in the description of the DRF algorithm. The failure probability of a path is given by the sum of the failure probabilities of the links along the path. The minimum failure probability between two nodes, mfp_{sd}, is the minimum between the failure probability of the clockwise and the counterclockwise path. Let $MFP(c)$ be the maximum failure probability for connection demands in class c.

The algorithm is organized in 7 steps as depicted in Fig. 3.

Step 1. Connection demands are classified using two sets: the set of demands that require some degree of protection and the set of demands that do not require protection. Demand of class c from node s to node d belongs to the former set if $mfp_{sd} > MFP(c)$. It belongs to the latter set otherwise.

Step 2. The working lightpath for each demand in the former set is routed using the shortest path in terms of number of links. The protection lightpath is routed using the opposite direction. Shortest path in terms of number of links is chosen for the working lightpath to yield the maximum number of protection wavelengths in the ring that may be reused by the algorithm at some later step. Notice that these connections have failure probability equal to 0, as they all survive any single fault in the ring.

Step 3. The demands in the latter set are sorted according to increasing values of the difference $X = (MFP(c) - mfp_{sd}) \geq 0$ where c, s and d are, respectively,

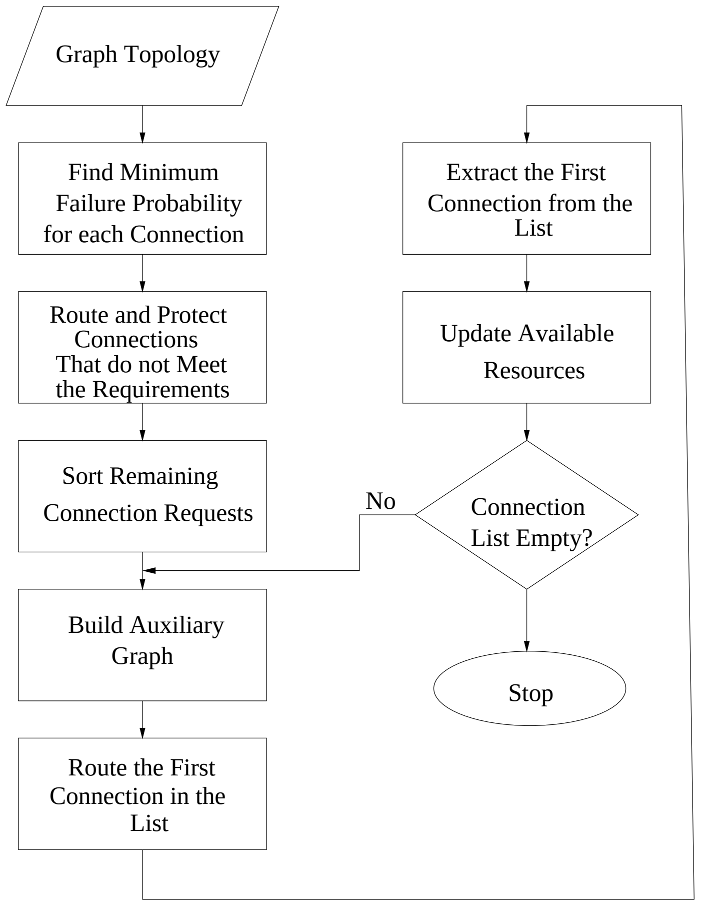

Graph Topology
Find Minimum Failure Probability for each Connection
Route and Protect Connections That do not Meet the Requirements
Sort Remaining Connection Requests
Build Auxiliary Graph
Route the First Connection in the List
Extract the First Connection from the List
Update Available Resources
Connection List Empty?
No
Stop

Fig. 3. Algorithm flow chart

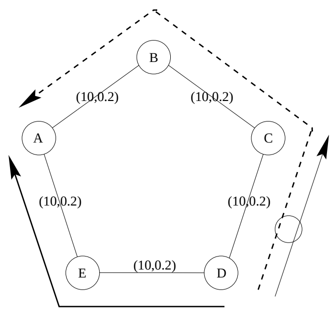
the class, the source and destination of the demand. Let S_X be the set of sorted demands. Value X indicates the *excess of reliability* offered to the demand if a newly wavelength was added to each link of the the path with minimum failure probability mfp_{sd}. Since the excess of reliability is not necessary, the algorithm looks for ways to reuse some of the already provisioned protection wavelengths in place of the newly added wavelengths. When a protection wavelength is used in place of a newly added wavelength, the reliability excess of the demand is reduced due to the potential preemption of that wavelength. Notice that the reliability excess must not be reduced below zero, otherwise the required reliability degree requested for the demand is not met.

Intuitively, smaller values of X correspond to demands with lower probability to achieve reuse of protection wavelengths. In order to achieve a more efficient wavelength reuse — which in turn corresponds to a lower total network cost — demands having smaller X are routed first, hence the name of the algorithm.

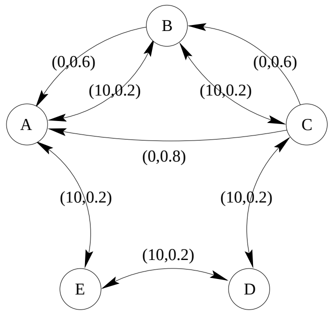
Step 4. The demand in set S_X with the smallest value of X is considered for routing. Prior to routing the demand, an auxiliary graph $G'(\mathcal{N}, E')$ is built. $\mathcal{N}$ is the set of network nodes, E' is the union of set $E \in G$ and set E_p. The latter is the set of links that represent the provisioned protection lightpaths (or segments of them) not yet reused. Set E_p is constructed as follows. Let s be the source node, d the destination node and $I = \{n_1, n_2, \ldots, n_k\}$ the ordered set of intermediate nodes of a protection lightpath. Link (s, d), all links (s, n_i), where $n_i \in I$, all links (n_i, d), where $n_i \in I$, and all links (n_i, n_j), where $n_i, n_j \in I$ and $i < j$ are added to E_p when the associated protection lightpath is added to the ring. Link (s, d) represents the entire protection lightpath. Any other link represents only a segment of the protection lightpath. When multiple protection lightpaths share the same source and destination pair the set of links is derived only once, taking into account the multiplicity of the lightpaths by assigning a capacity greater than one to the links.

Any link in $l \in E'$ is assigned a triple: a cost, a failure probability, and a capacity that indicates the maximum number of working lightpaths one can route on such link. Every link in E' derived from E is assigned its original cost, i.e., the length of its line, its original failure probability, and infinite capacity. Every link in E_p is assigned zero cost as routing on any such link represents reuse of an already provisioned protection lightpath (or segment of it). Each link in E_p is assigned a failure probability that is the sum of two terms: the failure probability of the working lightpath associated with the protection lightpath (or segment of it) represented by the link, i.e., the probability of preemption; and the failure probability of the represented protection lightpath (or segment of it). Each link in E_p is assigned a capacity equal to the number of protection lightpaths (or segment of lightpaths) represented by the link.

Fig. 4(a) represents the auxiliary graph obtained from the ring shown in Fig. 4(b). In Fig. 4(a) the original topology graph with two working connections — the solid lines — is shown. Links are assigned two numbers that represent, respectively, the length of the network line and the network line failure probability. It is assumed that connection demand from node D to node A requires protection, represented by the dotted line in the figure. Connection demand from node D to node C belongs to a lower requirement reliability class and thus it may reuse some of the protection wavelengths provisioned for the former demand. The corresponding auxiliary graph is shown in Fig. 4(b). Set E' consists of the union of link set E and the additional links (C, B), (B, A), and (C, A) (set E_p), that represent, respectively, the possibility to reuse protection wavelengths on the original graph links (C, B), (B, A), and the concatenation of (C, B) and (B, A). The auxiliary graph does not contain link (D, C), since the protection wavelength on that link is already used. Notice that added link (C, A), which represents a segment of the protection lightpath associated with the working lightpath from node D to node A, has failure probability given by the sum of three terms: the probability that the working lightpath from D to A is rerouted because of a line fault (0.4); the probability that line (C, B) is faulted (0.2); and the probability that line (B, A) is faulted (0.2).



(a) Original graph



(b) Auxiliary graph

Fig. 4. New links in the auxiliary graph

Since the auxiliary graph may have a large number of parallel links that may considerably slow down the execution time of the algorithm, a pruning technique is used to reduce the number of links in E', without significantly affecting the algorithm performance. Two steps are used to reduce the number of links in E'. First, links with failure probability greater than the demand MFP are removed from the auxiliary graph. Second, when multiple links with zero cost exist between a node pair, only the link with the smallest failure probability is kept in set E' (Fig. 5), the other zero cost links are not inserted in the auxiliary graph set E'. Notice that links of the auxiliary graph derived from E are never removed from the auxiliary graph as they represent the possibility to add wavelengths to the ring when the reuse of provisioned protection wavelengths is not possible.

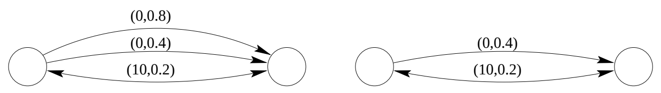


Fig. 5. Pruning of parallel multiple links with zero cost

Step 5. The demand under consideration is routed using the Dijkstra shortest path algorithm [5] applied to the pruned auxiliary graph. The link weight l^w_{ij} used by the shortest path is computed as a linear combination of the link length (l_{ij}) and the link failure probability $(P_f(i,j))$. The linear combination is controlled by parameter a as follows: $l^w_{ij} = a \cdot l_{ij} + (1-a) \cdot P_f(i,j)$. When $a = 1$ the algorithm returns the shortest path in terms of mileage, when $a = 0$ the algorithm returns the shortest path in terms of failure probability, i.e., the most reliable path.

By varying a, a dichotomic search is performed. Let $a_{max} = 1$ and $a_{min} = 0$. The shortest path algorithm is run using the mean value $a = a_{int} = (a_{max} + a_{min})/2$). Let $pf_{a_{int}}$ be the failure probability of the path so found. If $pf_{a_{int}}$ is greater than the MFP requested for the demand a_{max} is set equal to a_{int}. If $pf_{a_{int}}$ is smaller than the MFP, a_{min} is set equal to a_{int}. These steps are iterated until $pf_{a_{max}} \leq MFP$.

Once the path is found, the capacity of the path is computed as the minimum capacity of all links that belong to the path. Working lightpaths of the demand are then routed along the path, up to its capacity. If the capacity of the shortest path is not large enough to accommodate all lightpaths of the demand, Step 5 is repeated until all lightpaths are routed.
Step 6. The wavelengths assigned to the demand under analysis are provisioned and the corresponding demand is removed from set S_X.
Step 7. The status of the provisioned protection wavelengths is updated taking into account possible reuse. Notice that such update may result in changes of the auxiliary graph. The algorithm goes back to step 4 until set S_X is emptied.

3 Performance Results

The DRF algorithm described in section 2.2 is tested using a ring topology that comprises 20 nodes connected by equal length lines of 10 miles. It is assumed that line failure probability is uniformly distributed among the network lines, i.e., $P_f(i,j) = 1/20 \forall (i,j) \in E$. Connection demands are divided into three classes.

- Class 1 demands are uniform, requiring one lightpath between every node pair (s,d). $MFP(1) = p_1$.
- Class 2 demands are uniform, requiring two lightpaths between every node pair (s,d). $MFP(2) = p_2$.
- Class 3 demands are uniform, requiring three lightpaths between every node pair (s,d). $MFP(3) = p_3$.

The total number of lightpath requests is 2280. Presented results are obtained setting p_1 and p_2 to a fixed value, while p_3 takes on values in the interval $[0,1]$. With the given ring and connection demands, the DRF algorithm runs on a Linux-PC pentium $450MHz$, requiring a computational time in the order of tens of seconds.

Figs. 6, 7 and 8 show, respectively, the total λ-mileage (working and protection), the total protection λ-mileage, and the total protection λ-mileage reused by the DRF algorithm.

As expected, the required total λ-mileage decreases as the reliability requirement of class 3 becomes less stringent, i.e., p_3 increases. This is due to the combined effect of two factors: 1) the protection λ-mileage necessary to fulfill the requested reliability degree is reduced (Fig. 7); 2) protection wavelength reuse is improved (Fig. 8).

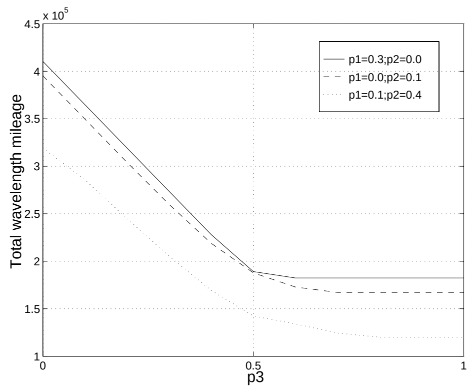


Fig. 6. Total wavelength mileage

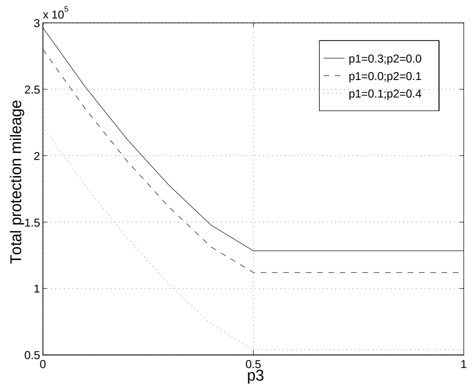


Fig. 7. Total protection mileage

It is interesting to note that when $p_1 = 0.1$, $p_2 = 0.4$ and $p_3 \rightarrow 1$, the total λ-mileage found by the DRF algorithm equals the same λ-mileage that would be required in the same ring, with the same connection demands, had conventional shortest paths been used to route the working lightpaths and no protection lightpaths been provisioned at all. While costing the same, the two approaches are however significantly different. With the shortest path approach the requested demand reliability is not always guaranteed and shorter working lightpaths have better reliability degree than longer lightpaths have. With the DRF algorithm, demands in the three classes always meet their reliability degree, without regard of the distance between the source and the destination.

Figs. 9, 10, and 11 numerically demonstrate the above conjecture by plotting the distribution of the connection failure probability for the demands in the three reliability classes when $p_1 = 0.1$, $p_2 = 0.4$, and $p_3 = 0.8$. The results obtained with the shortest path algorithm and the DRF algorithm are reported. The total

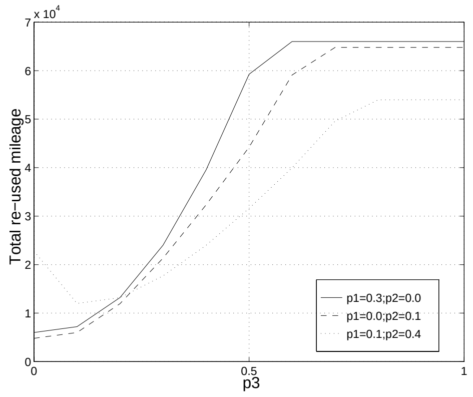


Fig. 8. Total reused mileage

λ-mileage is the same in both cases. The shortest path routing yields the same distribution of connection failure probability for every reliability class, irrespective of the reliability requirement of the class. (The distribution is actually scaled by a factor equal to the number of lightpath requests in each class.) The DRF algorithm trades higher connection failure probabilities for class 1 traffic, with lower connection failure probabilities for the other two classes. By so doing, the requested class reliability degree is met for all connection demands.

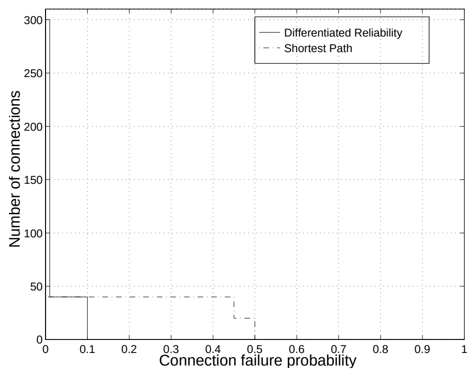


Fig. 9. Class 1 connection failure probability distribution

From a revenue viewpoint, the DiR approach allows to accommodate all connection demands with the requested degree of reliability. The conventional shortest path approach, on the contrary, imposes a de-classification of a large portion of the demands in class 1 and class 2 below the requested reliability

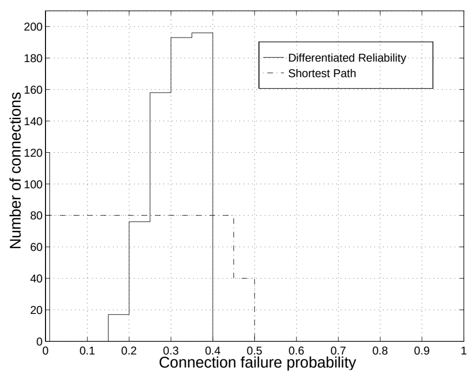


Fig. 10. Class 2 connection failure probability distribution

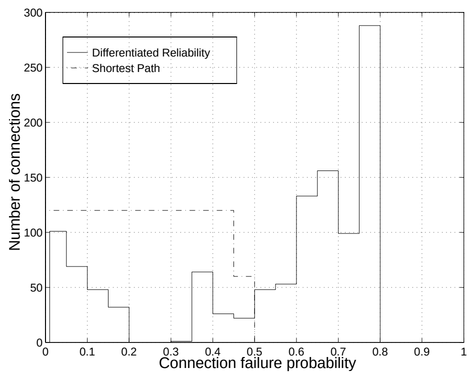


Fig. 11. Class 3 connection failure probability distribution

degree, thus reducing the potential overall revenues, yet requiring the same λ-mileage of the DiR approach.

4 Summary

The paper presented the novel concept of Differentiated Reliability (DiR) classes of traffic, according to which, connections are differentiated based on their requested individual degree of reliability. The proposed concept enables to differentiate traffic flows in classes without regard for network topology, equipment MTBF, and most importantly connection span, both in terms of line hops and mileage.

An algorithm was proposed to sub-optimally design DiR-based optical rings. Presented results demonstrated the potential advantages of the proposed concept, in terms of both overall network costs and guaranteed reliability degree for every traffic class. Beside the optical layer discussed in the paper, the DiR concept may find other interesting applications in a variety of network layers, including the Multi Protocol Label Switching (MPLS) layer.

References

1. D. Cavendish. Evolution of optical transport technologies: From SONET/SDH to WDM. *IEEE Communications Magazine*, June 2000.
2. P. Demeester, M. Gryseels, A. Autenrieth, C. Brianza, L. Castagna, G. Signorelli, R. Clemente, M. Ravera, A. Lajszczyk, D. Janukowicz, K. Van Doorselaere, and Y. Harada. Resilience in multilayer networks. *IEEE Communications Magazine*, 37, No 8, August 1999.
3. P. Demeester, M. Gryseels, K. Van Doorselaere, A. Autenrieth, C. Brianza, G. Signorelli, R. Clemente, M. Ravera, A. Jajszcyk, A. Geyssens, and Y. Harada. Network resilience strategies in SDH/WDM multilayer networks. In 24th *European Conference on Optical Communication (ECOC) '98*, volume 1, pages 579–80, Madrid, 1998.
4. N. Golmie, T.D. Ndousse, and D.H Sue. A differentiated optical services model for WDM networks. *IEEE Communications Magazine*, 38, No 2, February 2000.
5. M. Gondran and M. Minoux. *Graphs and algorithms.* Wiley Interscience, 1979.
6. IEEE Communications Society. *IEEE Communication Magazine*, volume 37, No 8, August 1999. Special Issue on Survivable Communication Networks.
7. IEEE Communications Society. *IEEE Network*, volume 13, No 5, September/October 1999. Special Issue on Integrated and Differentiated Services for the Internet.
8. A. Jukan, A. Monitzer, and H. R. Van As. QoS-restorability in optical networks. In *Proceedings of* 24th *European Conference on Optical Communication (ECOC'98)*, volume 1, pages 711–712, 1998.
9. S. Makan, V. Sharma, K. Owens, and C. Huang. Protection/restoration of MPLS networks. draft-makam-mpls-protection-00.txt, Expiration day April 2000, IETF, October 1999.
10. M. Ajmone Marsan, A. Bianco, E. Leonardi, A. Morabito, and F. Neri. All-optical WDM multi-rings with differentiated QoS. *IEEE Communications Magazine*, 37, No 2, February 1999.
11. R. Ramaswami and K. N. Sivarajan. *Optical Networks: a pratical prospective.* Morgan Kaufmann Publishers, Inc., 1998.
12. R.W. Stevens. *TCP/IP Illustrated, Volume 1*, volume 1. Addison-Wesley Professional Computing Series, 1994.
13. T.H. Wu. *Fiber Optic Survivability.* Artech House, 1992.

An Optical Packet Switch for IP Traffic with QoS Provisioning

Franco Callegati, Giorgio Corazza, and Carla Raffaelli

D.E.I.S., University of Bologna, viale Risorgimento 2
I-40136 Bologna, ITALY
{fcallegati,gcorazza,craffaelli}@deis.unibo.it

Abstract. This paper addresses the problem of building optical packet switches able to effectively cope with variable length packet traffic, such as IP traffic. A switching architecture equipped with a multistage fiber delay lines based buffer is presented. The aim is to realize a buffer with fine granularity and long delay with an architecture of limited complexity. QoS mechanisms are proposed to support service differentiation by exploiting the wavelength resource and are evaluated by simulation.

1 Introduction

The explosive growth of the Internet of the last few years demands for higher and higher bandwidth capacity, in particular in the core part of the network. The recent and rapid introduction of Dense Wavelength Division Multiplexing (DWDM) technology provides a platform to exploit the potential huge capacity of the optical fibers. DWDM optical networks are poised to dominate the backbone infrastructure supporting the next-generation high-speed Internet backbones. They offer a very effective and future proof transmission facility, that may be used to support a variety of services and transfer modes in a fairly transparent and smooth way.

Nevertheless, current applications of WDM focus on the static usage of individual WDM channels, which may not be efficient in supporting IP traffic. The challenge is to combine the advantages of WDM with emerging all-optical packet switching capabilities to yield optical routers capable of guaranteeing full optical data path and throughput in the hundreds of Terabit/s range [2][1].

Optical packet switching has been addressed by several projects in the last few years showing the feasibility of such a concept but also its partial immaturity at the present state of optical technology development [3],[4],[5]. These works mainly considered an ATM like transfer mode, with fixed-length packets and synchronous node operation. Nevertheless the present scenario of a network development driven by the Internet (meaning IP protocol) and the difficulty in realizing full-optical synchronization seems to suggest that a transfer mode based on variable length packets and asynchronous node operation may be easier to implement and more effective. Some proposal have been done in this direction [6][7] but very limited study is present in the literature on node architectures and related performance.

M. Ajmone Marsan, A. Bianco (Eds.): QoS-IP 2001, LNCS 1989, pp. 315–330, 2001.

This paper discusses some of the engineering and dimensioning issues of optical packet switches in the case of asynchronous variable-length packets and proposes a node architecture suitable for this networking environment. The main features of this architecture are a multistage fiber delay line buffer, similar to the one proposed in [8] for the case of fixed length synchronous packets, and WDM operation to solve contention on the use of the fiber delay lines by several packets at the same time. Results will show that, by means of such a buffer, very good performance in terms of packet loss are obtained, without a dramatic increase in the architecture complexity. In relation to the increasing need to obtain different service levels from the network, the design freedom degrees of the architecture are also investigated with the aim to define mechanisms to support Quality of service (QoS) features. The paper is structured as follows. In section 2, a general discussion of the scenario of optical packet switching is presented. In section 3 the switching architecture proposed is explained in detail, with focus on the functional behaviour. In section 4 the main design issues are discussed with reference to the play they role in determining the system performance. In section 5 the basic switch performance are presented. In section 6 quality of service mechanisms are described and their performance discussed. In section 7 conclusions are drawn.

2 Optical Packet Switching Scenario

The idea underlining any proposal for optical packet switching is to de-couple the data-path from the control path. Since a device where processing is fully performed in optics is not feasible yet, the control functions for routing and forwarding have to be performed in electronics. This implies optical to electrical conversion for the packet header (whatever the format). This is not necessary for the packet payload, that may be switched and forwarded without conversion. If feasible and efficient, this approach may be a significant improvement when compared, for instance, with the SDH approach, where any information flow must be converted in electronic form, de-multiplexed, processed, multiplexed again and finally re-converted to optical at each cross-connect.

Optical packet switching promises to be the next step forward in switches and routers development, even though it is not clear yet when it will really be effective. We will simply assume that optical packet switching will be feasible and worth doing at some stage in the future.

Based on this assumption, the discussion that follows wants to introduce the scenario on which the architecture and results here presented are placed. The network architecture considered is a packet switched network with optical core routers in the backbone and optical edge routers, responsible for the interface with legacy networks at the edges. The traffic may in principle be of any kind even though we imagine Internet traffic to be dominant. The backbone links are WDM links at 2.4 Gbit/s or higher and the information transfer mode is packet based, to allow a good degree of multiplexing and flexibility.

As far as the packet format is considered the possible Approaches are:

- fixed length packets (tailored on the time requirement of the system) with synchronous, ATM like node operation [5] [1];
- fixed length packets with asynchronous operation [14];
- variable length packets, called bursts, with asynchronous node operation [6].

The synchronous operation of the former approach is very appealing as far as switches implementation is concerned, at least in the case a connection oriented switching is assumed. Unfortunately it is not very well tailored to IP traffic, both because of the native connectionless nature of IP and because of the necessity to fit variable length datagrams into fixed length containers. In particular regarding the latter issue, it has been shown in [15] that it may be very inefficient. The fixed length asynchronous case may simplify some of the implementation issues related to synchronisation but leads to fairly poor performance in terms of packet loss.

The asynchronous variable length packet, in particular the so-called *burst switching* approach, first proposed in [6], definitely matches better with IP traffic. In burst switching the IP datagrams are gathered in "bursts" to guaranty a pre-defined minimum length and bursts are switched in a connectionless manner according to the destination address in the header.

This is the transfer mode we assume in this paper. We analyze in some detail which are the differences with the case of fixed length packets from the switch operation point of view and propose a switching architecture for this case. The focus is on the buffering problem.

As in any packet switch congestion may arise when more than one packet are addressed to the same output at the same time. To solve this contention some kind of buffering is necessary. Up to now the most common way to realize an optical buffer has been by means of fiber delay lines (FDLs). When two (or more) packets contend for the same output link, one is transmitted and the other is sent to a coil of fiber to be delayed of an amount of time sufficient to solve the contention. There are several basic parameters that can be used to characterize an FDL buffer. We will call:

- D the basic delay unit, also called "granularity";
- B the number of different delays that can be realized overall;
- $D_M = BD$ the maximum delay achievable with the buffer, that also gives the maximum amount of bits that may be stored in the buffer, for this reason also called "buffering capacity";
- L the number of wavelengths used inside the FDL buffer, necessary in order to share the same pool of fibers among all the input/output pairs.

D and L are the critical design parameters for the FDL buffer. In the case of a network with fixed lengths packets (synchronous or asynchronous) it is natural to choose D equal to the length of the packets and results regarding dimensioning of FDL buffers and congestion resolution techniques in this case can be found for instance in [9],[10],[11]. On the other hand, very few work has been devoted to the problem of dimensioning FDL buffers in the case of networks with asynchronous variable length packets (see [12] as one of the very few examples).

3 Switch Architecture

The packet switch architecture realizes output queuing of and relies on the use of wavelength encoding to achieve optimal exploitation of the fiber delay line buffer. For the sake of simplicity in the description each input link is assumed to carry only one wavelength, but the extension to the WDM scenario is straightforward. The switch consists of four main functional blocks (figure 1):

– the packet encoder, which consists of a set of L wavelength converters to access the delay stage in a contention free manner;
– the variable delay stage, which provides multiple stages of delay lines to introduce variable delays;
– the wavelength selector, to forward a specific wavelength to the addressed output, by selecting the correct wavelength;
– the control block, which performs the routing function and controls wavelength assignment and packet forwarding.

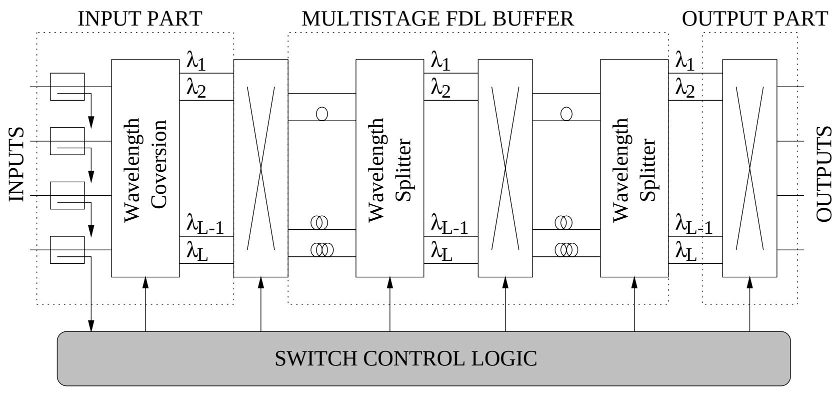


Fig. 1. Switch architecture

The packet encoder. When a packet arrives on a switch input link its header is read by means of an optical detector and processed to determine the switch output to which the packet must be forwarded. On the optical path each packet is assigned to a wavelength chosen such that no other packet is using it or is going to use it for the whole stay of the new packet in the delay stage. Different algorithms can be adopted to achieve optimal use of the wavelength pool, in any case they must be kept simple enough to cope with the very high speed of the switch. In this work it is assumed to choose the first wavelength that satisfy the constraints starting from a fixed one (for example the first one). The encoding wavelength is maintained for the packet during the whole switch path.

The variable delay stage. After encoding, the packet enters the delay stage and flows into the fibers of the proper length to be delayed of the required amount. A multistage configuration of the delay lines is proposed to achieve a fined-grained implementation of delay. A base m representation of packet delay is given through the k stages, with stage weights in decreasing order. For a given k, the i-th stage is composed by m fibers that introduce the following multiples of thedelay unit D: 0, m^{k-i}, $2m^{k-i}$, $3m^{k-i}$, $\ldots$, $(m-1)m^{k-i}$. By properly choosing the delay line at each stage all possible delays between 0 and $D_M = Dm^{k-1}$ are possible. To achieve the correct composition of the delay contribution, each delay stage is followed by a combining/splitting function, a pool of filters that separate wavelength encoded packets and a space switch to send a packet to the correct FDL in the following stage. The connections between the combiners and the splitters are the critical points of the architecture where it must be assured that no packet jamming takes place on the same wavelength. At this connections only one packet at a time can be present on each given wavelength and this is taken into account by the wavelength encoding algorithm. The values of m and k must be chosen in relation to the performance that has to be obtained. They are limited by technological constraints and their influence on performance is here investigated.

The wavelength selector let the packets exiting the right switch output according to control information by means of a pool of optical filters and a space switch.

The control block. The control block provides for the routing function and management of switch resources. The routing function is performed on the basis of the detection of packet header using standard Internet routing procedure and should be performed suitably in advance before packet information arrival. The result of the routing function is the specific output link to which the packet must be forwarded. This also implies the knowledge of the delay that the packet will encounter in the switch, based on the status of the output queue. The next step is to find the wavelength to be used for packet encoding such that the required delay, that is the FDL buffer, is available at that wavelength for the whole packet duration T_P.

The implementation of this architecture relies on key devices such as All Optical Wavelength Converters and Silicon Optical Amplifiers (SOA) gates for the optical buffer, the space switches and the Wavelength Selectors. The feasibility of the basic components necessary for this architecture has been already proved for similar applications [10]. Even if they are not yet available on the mass market it is reasonable to think they will in the next few years.

As a final remark, according to the scheme in figure 1 packets on the same output link could be encoded on different wavelengths. To send all the packets on a given output on the same wavelength (that is the wavelength used by the link) a final stage of wavelength conversion is needed. It has not been shown here, since it is not vital for the switching matrix operation.

4 Design Issues

In this section we discuss the design issues related to the architecture presented. Since the focus is on buffering and loss of packets because of lack of queuing space, the discussion is centered on explaining the functional behavior of the buffering achieved by the architecture and, consequently, the reasons of packet loss.

4.1 FDL Buffers and Variable Length Packets

Let us first analyze the delay assignment. If a packet arrives and the output link is idle it is obviously served immediately. If a packet arrives and finds the link busy or other packets queued, it has to be buffered (i.e. delayed). Call t_a the time of the arrival and t_f the time at which the link will be free. In principle the new packet should be delayed of an amount $t_f - t_a$. Due to the discrete step of the FDLs, the new packet is going to be delayed of an amount

$$\Delta = \left\lceil \frac{t_f - t_a}{D} \right\rceil D$$

In the case of the architecture here described, Δ will be realized by means of a suitable choice of delay in the various stages. If $[d_1 \dots d_k] \quad 0 \leq d_i \leq m-1$ is the vector of the fiber delay line indexes to which the packet will be sent at each stage, then $\Delta = \sum_{i=1}^{k} d_i m^{k-i} D$.

In general there is is an amount of time $\tau = \Delta - t_f + t_a \geq 0$ during which the output line is not used while there would be a packet to transmit.[1] This can be seen as a waste of some of the "service power" of the output line or as an artificial increase in the packet length. If ϑ is the nominal length of a packet then, for any packet queued, $\vartheta' = \tau + \vartheta$ is the real length.[2] In the following this last way to represent the effect of the FDLs is used. Let us assume that the arrival process is independent of the state of the queue and that the lengths of consecutive packets are also independent. When a new packet arrives at t and has to be queued, τ may take any value between 0 and D with uniform probability. If $\bar{\vartheta}$ is the average nominal length of the packets, the real average length for the packets queued is $\bar{\vartheta}' = \bar{\vartheta} + \frac{D}{2}$.

If ρ is the load for normal queue the real load in the case of the FDL buffer is $\rho_r \geq \rho$ because a fraction of the packets have an increased length. We can write $\rho_r = \rho + \rho_e$ that put in evidence the additional contributions $\rho_e \geq 0$, called *excess utilization* [12], that is related to the discretization of delays introduced by the FDL buffer.

[1] Note that it may happen that an incoming packet is queued even if the link is idle. This happens when one packet has been served and there is one packet already queued but not served yet since its delay has not expired.

[2] Let us define the length of a packet as the amount of time the output link is busy because of such packet, even if for τ nothing is really transmitted.

As a result of this discussion, intuition suggests that, given a certain B (fixed m and k), there must be a sort of optimal value for the buffer granularity, that can be explained as follows:

– if D is very small (in the limit going to 0) the time resolution of the FDL buffer increases but the buffering capacity D_M also decreases (in the limit there is no buffering capacity if D approaches 0), therefore the performance in terms of packet loss must improve increasing D,
– if D is very large (in the limit going to infinity), D_M is large, long delays can be introduced, but the time resolution of the buffer is very small, because of the large granularity, and the excess utilization is large (in the limit going to infinity with D), therefore the performance in terms of packet loss must improve decreasing D.

In between these two situations we expect a sort of minimum of the packet loss, realizing the optimal trade off between time resolution and amount of delay achievable. The existence of such a minimum has been shown by means of simulation in [12].

4.2 Exploitation of WDM

In order to achieve hardware optimization and reduce the splitting/recombining factor inside the architecture to feasible values, WDM is used inside FDL buffers. Since a unique set of FDL, shared among the whole paths inside the switch, is provided at each buffering stage, WDM is necessary to avoid packet jamming when more packets overlap in time, directed to different outputs. Obviously packets travelling the same delay line but to different outputs must be multiplexed in the wavelength domain. This seems to suggest that two packets travelling different delay lines at the same time could be encoded at the same wavelength. This could be in principle but is not in this architecture. This because the most critical points are the connection fibers between each splitter/combiner pair downstream each delay stage. In this points all the packet crossing the switch in a given instant are merged, therefore they must all be encoded on different wavelength.

For each incoming packet the control algorithm searchs for a suitable wavelength. The choice is done by searching a wavelength that is not already used in any of the delay lines crossed by the packet. If $t_i = \sum_{i=1}^{j-1} d_i m^{k-1} D$ is the time a packet enters the proper fiber delay line at stage j, then it may be encoded at a given wavelength if this wavelength is not used in the interval $[t+t_i, T+t_i+T_p]$, being T_p the length of the packet. A wavelength is available again when the packet has been completely transmitted.

As a result of the asynchronous arrival of packets and of statistical behavior of their duration in time, the wavelength assignment produces a discontinuous use of the wavelengths that alternates busy and idle periods of random length. Two main different strategies can be adopted if no wavelength is available to meet the time requirements of a packet: a drop-oriented strategy that simply discard the packet and a delay-oriented one that try to assign the packet a delay greater than Δ. In this paper the first one is analyzed with the aim to maintain the assignment procedure as simpler as possible.

4.3 Packet Loss

Packet loss is the result of two combined effects:

- lack of wavelengths to encode the packet without jamming;
- lack of buffering space (meaning that the delay that would be necessary is larger than D_M).

Even though it is difficult to separate completely the two effects, we expect that, for small values of L the packet loss probability is mainly due to lack of wavelength to encode the packets. In this case the packet loss should improve increasing L with fixed D, m and k. We also expect some sort of saturation, meaning that at a certain point a further increase of L will not affect the packet loss probability any more. If this happen we would say that the packet loss probability is then just due to the lack of buffering capacity, meaning that the packets are lost because the required delay is not achievable even if there would be a wavelength available.

5 Basic Switch Performance

Performance of the described architecture has been evaluated by means of an event driven simulator that performs calculation of very low loss probabilities (e.g. 10^{-7}) in a few minutes with very good confidence (10^{10} generated packets) on commercially available PCs. The analysis has been developed in relation to the typical parameters of the architecture that are the number of wavelengths, the number of stages and the counting base, these two determining D_M.

The main performance figure is the packet loss probability, being it the most critical due to the limited optical buffer capacity. A 4×4 switch is considered with offered load equal to 0.8 and average packet length 500 byte. Input traffic is generated as the output of M/M/1 queues, that represents the simplest ideal model for variable length packet traffic. The granularity D has been chosen as the independent variable for plotting curves to evidence its influence on performance in a design perspective. As expected all curves exhibit a minimum representing the optimal trade off.

Figure 2 show the two-stage solution with base $m = 16$, each curve corresponding to a different value of the number of wavelengths. Increasing L leads to better performance as long as the loss due to the lack of wavelength is the dominant effect. A maximum value of L exists over which the loss probability is only related to the limited output queue size: so increasing L has no more effect on performance.

By adding another delay stage as shown in figure 3 performance enhancement is obtained due to the increase of buffer capacity (same D but bigger k). Simulations have shown that in a large range of values of D, a packet loss probability lower than 10^{-8} is achievable with $L = 14$. In any case the number of stages is limited for technological reasons to few units [16].

Another system parameter is represented by the counting base. In figure 4 performance comparison is provided for two cases: $m = 8$ and $m = 16$ with $L = 12$ wavelengths. The use of a larger base allows for deeper buffer and leads to lower packet loss probability. Moreover the granularity D can be kept lower such that better performance in terms of packet delay can be achieved.

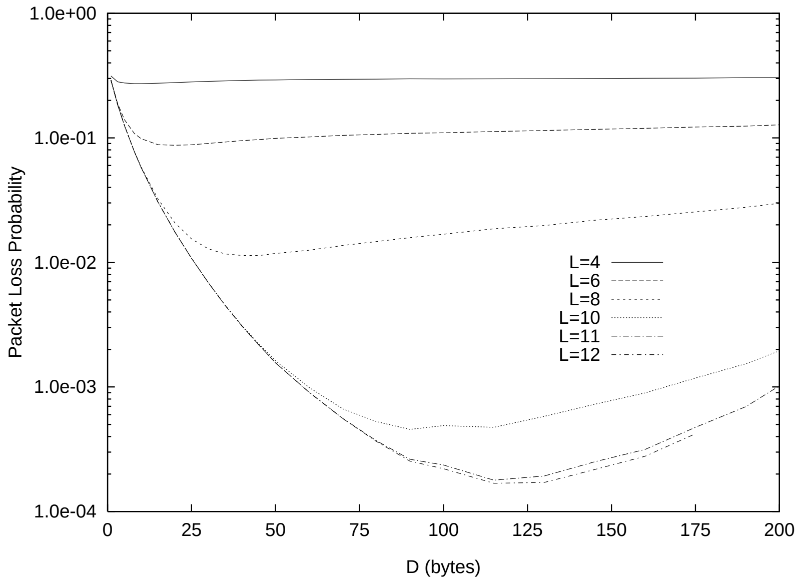


Fig. 2. Packet loss probability as a function of the delay granularity D, for a switch with a 2 stage buffer, base 16, varying the number of wavelengths as a parameter

6 Support of Service Differentiation

Service differentiation is becoming a very important issue in the Internet as a consequence of the variety of applications and the related growing amount of traffic produced. To this end an intensive research and standardization effort has taken place in the last few years in order to define the most suitable paradigm for service differentiation support [22]. This problem has been also considered for the optical packet switching domain, where, in spite of the huge bandwidth available on optical links, congestion may in any case occur in the queues of the nodes, whose length is typically limited for technological reasons [11] [10]. The support of different levels of quality of service has been studied for the proposed switch

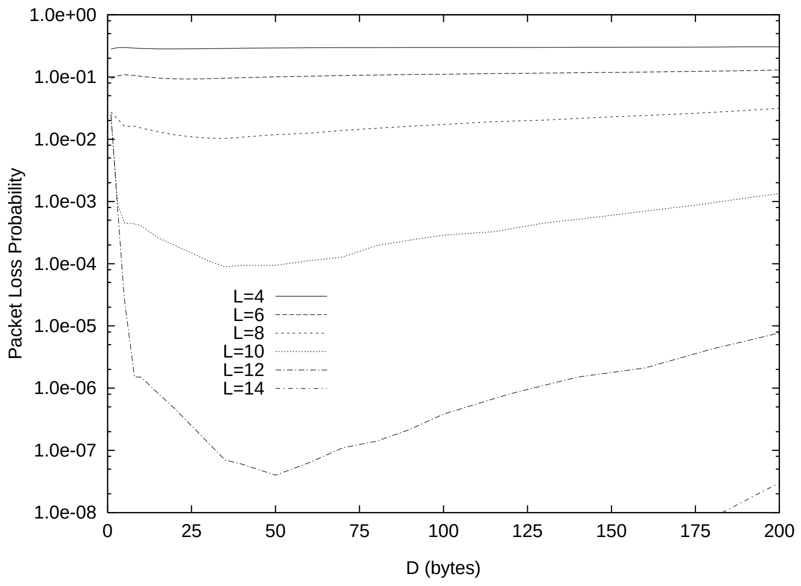


Fig. 3. Packet loss probability as a function of the delay granularity D, for a switch with a 3 stage buffer, base 16, varying the number of wavelengths as a parameter

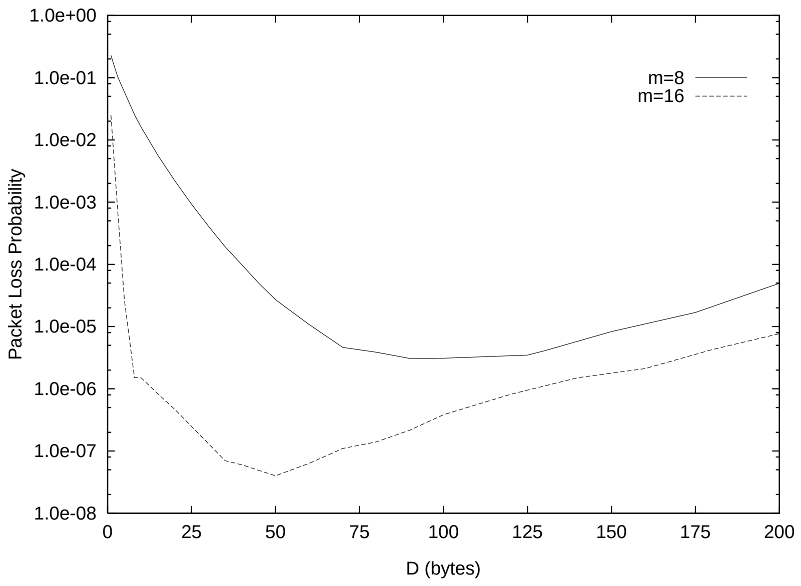


Fig. 4. Packet loss probability as a function of the delay granularity D, for a switch with a 3 stage buffer, varying the buffer base and the number of wavelengths

architecture with reference to the differentiated service model proposed by IETF [17], that seems to be the most suitable and scalable solution especially for the high capacity optical envirinment. Within the differentiated service paradigm two main node behaviours have been defined to achieve service differentiation, that is the assured forwarding (AF) [19] and the expedited forwarding (EF) [18] per hop behaviours: here the AF is considered and solutions to support three levels of service, in terms of packet loss probability, within a single AF class are proposed. The three level of service are thought to correspond to high priority in profile traffic, high priority out profile traffic and best effort traffic in a decreasing priority order.

Different techniques can be implemented in a node to obtain service differentiation, like threshold dropping or priority scheduling in the buffers [20], [21]. In an optical packet switch node a further opportunity is given by the exploitation of the wavelength domain that is the main aspect that is here investigated. In the optical switch two different causes of congestion must be considered in the definition of a service differentiation mechanism: the limitation of buffer length and the limitation of the wavelength resource. This last in its turn is influenced both by the number of wavelength converters in the input stage and by the scheduling policing adopted for the wavelength encoding of the optical packet. Depending on the congestion cause, three different techniques are proposed and evaluated:

- *Wavelength Allocation (WA)*: this technique is used when the limitation in the number of wavelengths is the major contribution to packet loss. A different number of wavelength converters is assigned to each service levels to obtain service differentiation. In particular the lowest priority traffic sees the smallest subset of wavelength converters, beginning from the top first, the medium priority traffic sees a more greater subset and the highest priority traffic see all the wavelength converters. Some wavelength converters are thus shared among all the traffic levels, someone are shared among the two highest priority levels and a smaller subset is dedicated to the highest priority traffic level. By numbering the wavelength converters from the top, N_L is the number of wavelength converters shared among all service levels and seen by the lowest priority traffic, N_M is the number seen by the medium priority level and N_H is the number seen by the highest priority traffic: obviously $N_L < N_M < N_H = L$, the total number of wavelength converters.
- *Combined Wavelength Allocation and Threshold Dropping (WA/TD)*: this technique is used when the packet loss is given both by buffer limitation and wavelength unavailability in the same order of magnitude. It is based on a first two-level differentiation obtained at the wavelength converters by assigning at the highest priority levels all the input wavelength converters ant a small subset to the remaining two classes. A further differentiation is achieved among the two lowest priority classes in the buffer where a threshold is introduced: when the buffer occupation is above the threshold, the lowest priority packets are discarded while the other packets are accepted until the buffer is full. So service differentiation is achieved in the wavelength domain

for the two highest priority classes and in the buffer domain for the two lowest priority ones.

– *Wavelength allocation with Scheduling (WAS)*: this technique extends the wavelength allocation technique by exploiting also the wavelength encoding algorithm to achieve service differentiation. As in WA a different number of wavelength converters is assigned to each service level. Moreover a delay based scheduling is applied to the highest priority levels. If an highest priority packet does not find a wavelength to exactly obtain the required output queuing delay, its delay is incremented and the search repeated. If again an interval is not found a further delay is added and a third search is performed after which, if it is not successful, the packet is discarded. This mechanism is applied also to the second level traffic with only one further delay. The lowest priority traffic, in the case of wavelength unavailability, is discarded without any further delay.

Performance have been evaluated by simulation for the three techniques with the aim to obtain practically zero loss for the highest priority traffic and a suitable difference between the packet loss probability for the remaining levels. The switch is considered with a single stage with a buffer consisting of 512 FDLs and average packet length equal to 500 bytes. For the purpose of these evaluations the target packet loss probability less than $10^{-6}$is assumed for the highest priority traffic and the target difference between the two lowest levels is fixed at least equal to one order of magnitude. In figure 5 packet loss probability is plotted as a function of the granularity for the three levels of service with the WA technique, with $N_L = 20$, $N_M = 22$, $N_H = 24$. It can be seen a good behaviour in relation to the requirements, that can be optimised with respect to the granularity. The same set up for wavelength allocation is chosen for the WAS technique where scheduling delays equal to half the packet length and to the whole packet length are applied to the highest priority traffic and a delay equal to half the packet length is applied to medium priority traffic. The benefits of these WA enhancements are shown in figure 6 and allow better switch dimensioning and performance at the expense of more complicated control algorithm. The WA/TD has been evaluated for coarser time granularity when the buffer gives a not negligible contribution to packet loss. Here 22 wavelengths are seen by the two lowest priorities and all the 24 wavelength are seen by the highest priority traffic. It is evident in figure 7 that in order to maintain the packet loss for the high priority traffic low, it is more difficult to obtain sufficient differentiation between the lowest levels. So it can be concluded that the most effective solution for switch dimensioning in a quality of service perspective is based on a suitable management of the wavelength allocation and packet scheduling such that the required level of differentiation can be achieved without imposing thresholds on buffer occupancy. It is worthwhile noting that the mechanisms proposed are peculiar of the optical domain in the sense of exploiting the wavelength resource.

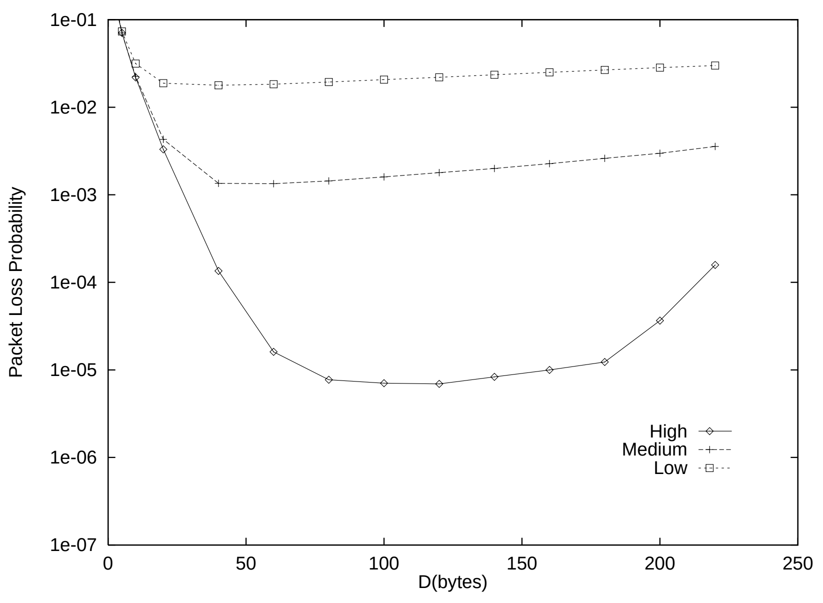


Fig. 5. Packet loss probability as a function of time granularity for the WA technique with N_L=20, N_M=22, N_H=24.

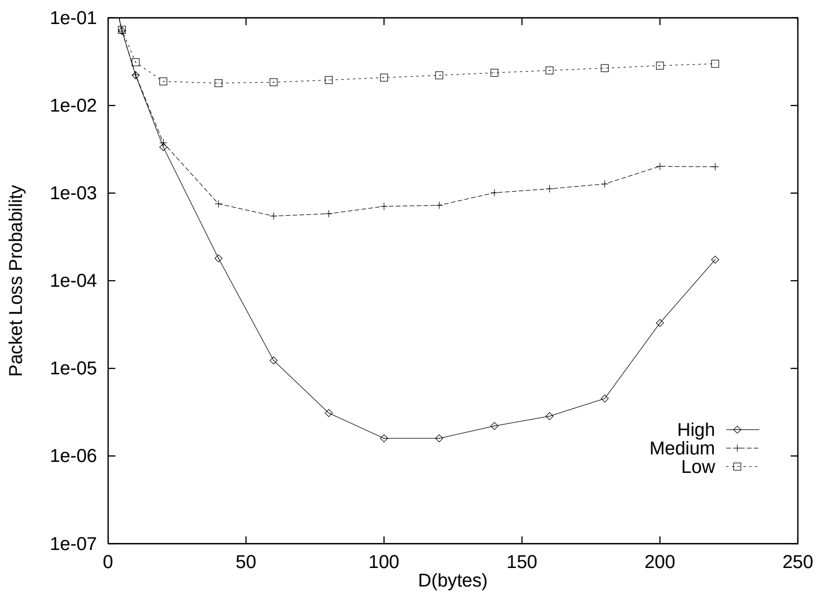


Fig. 6. Packet loss probability as a function of time granularity for the WAS technique with N_L=20, N_M=22, N_H=24.

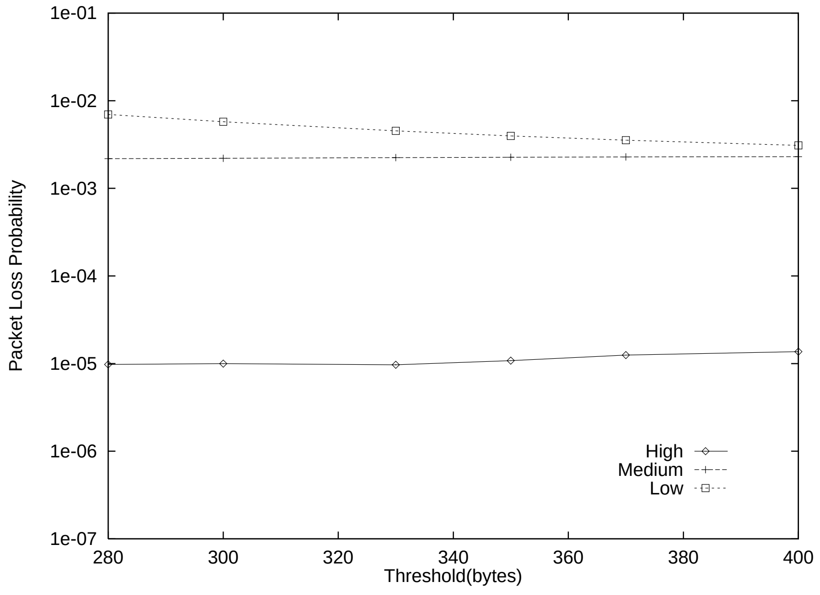


Fig. 7. Packet loss probability as a function of threshold position for the WA/TD technique with with N_L=N_M=22, N_H=24 time granularity equal to 45

7 Conclusions

In this paper a new switch architecture for variable length packets has been presented. The architecture performs output queuing of variable length packets using a multistage FDL buffers. It relies on feasible optical devices suitably interconnected. The main design parameters of the architecture are the number of delay stages, the FDL lengths and the number of wavelengths whose impact on packet loss has been analyzed in relation to typical environment conditions. The main results, obtained by means of an efficient event driven simulator, show that very low packet loss probability can be reached, with a suitable choice of system parameters. In particular it is crucial a correct dimensioning of the time granularity as well as of the number of wavelengths used to multiplex packet in the FDL stages. The QoS problem has also been investigated and mechanism that exploit the wavelength resources have been shown to be effective. Therefore the architecture has been demonstrated to perform accordingly to typical data network requirements and seems a very promising solution for photonic packet switching.

Acknowledgement

The authors wish to thank Mr. Daniele Pagliarani and Mrs. Francesca De Angeli for their help in implementing the simulation programs.

References

1. B. Bostica, F. Callegati, M. Casoni, C. Raffaelli, “Packet Optical Networks for High Speed TCP-IP Backbones”, *IEEE Communication Magazine*, Vol. 37, No. 1, pp. 124-129, January 1999.
2. T.W. Chung, J. Coulter, J. Fitchett, S. Mokbel, B.S. Arnaud, “Architectural and engineering issues for building an optical Internet”, Draft report, http://www.canare.ca/bstarn/optical-internet.html, July 1998.
3. F. Masetti et al., “High speed, high capacity ATM optical switches for future telecommunication transport networks”, *IEEE Jornal on Selected Areas in Communications*, Vol. 14, No.5, pp. 979-999, 1996.
4. L. Chlamtac et. al., “CORD: contention resolution by delay lines”, *IEEE Jornal on Selected Areas in Communications*,, Vol. 14, No. 5, pp. 1014-1029, 1996.
5. P. Gambini et al., “Transparent optical packet switching: network architecture and demonstrators in the KEOPS project”, *IEEE Jornal on Selected Areas in Communications*, Invited paper, Vol. 16, No. 7, pp. 1245-1259, 1998.
6. J. Turner, “Terabit Burst Switching”, www.arl.wustl.edu/ jst/, July 1998.
7. M. Yoo, M. Jeong, C. Qiao, “A High Speed Protocol for Bursty Traffic in Optical Networks”, SPIE Symposium on Broadband Networking Technologies, Dallas, USA, November 1997.
8. D.K. Hunter, W.D. Cornwell, T.H. Gilfedder, A. Franzen, I. Andonovic, “SLOB: A switch with large optical buffers for packet switching”, *IEEE/OSA Journal on Lightwave Technology*, Vol. 16, No. 10, pp. 1725-1736, 1998.
9. Z. Haas, “The Staggering Switch: an Elecronically Controlled Optical Packet Switch”, *IEEE/OSA Journal on Lightwave Technology*, Vol. 11, No. 5/6, pp. 925-936, 1993.
10. F. Callegati, M. Casoni, G. Corazza, C. Raffaelli, D. Chiaroni, F. Masetti, M. Sotom, “Architecture and Performance of a Broadcast and Select Photonic Switch” , *Optical Fiber Technology*, No. 4, pp. 266-284, 1998.
11. C. Guillemot et al., “Transparent Optical Packet Switching: the European ACTS KEOPS project approach”, *IEEE/OSA Journal on Lightwave Technology*, Vol. 16, No. 12, pp. 2117-2134, 1998.
12. L. Tancevski, A. Ge, G. Castanon, L.S. Tamil, “A New Scheduling Algorithm for Asyncronous, Variable Length IP Traffic Incorporating Void Filling”, IEEE/OSA OFC ’99, San Diego, Cal., March 1999.
13. L. Tancevski, L.S. Tamil, F. Callegati “Non-Degenerate Buffers: a paradigm for Building Large Optical Memories”, *IEEE Photonic Technology Letters*, 1999.
14. P.B. Hansen, S.L. Danielsen, K.E. Stubkjaer, “Optical Packet Switching Without Packet Alignment”, Proc. ECOC ’98, Madrid, Spain, September 1998.
15. F. Callegati, C. Raffaelli, “End-to-end delay evaluation for an Optical Transparent Packet Network”, Photonic Network Communications, Vol.1, pp 147-160,1999, Kluwer Academic Publishers.
16. D. Chiaroni et al. “Theoretical Feasibility Analysis of a 256x256 ATM Optical Switch for Broadband Applications”, ECOC ’93, Montreaux, Switzerland, Sep. 1993, pp. 485-488.
17. S. Blake, D. Black, M. Carlson, E. Davies, Z. Wang, Z. W. Weiss, ”An Architecture for Differentiated Services”, RFC 2475.
18. V. Jacobson, K. Nichols, K. Poduri, ”An Expedited Forwarding PHB”, RFC 2598.
19. J. Heinanen, F. Baker, W. Weiss, J. Wroclawski, ”Assured Forwarding PHB”, RFC 2597.

20. R. Rnngren, R. Ayani: "A comparative study of parallel and sequential priority queue algorithms", ACM Trans. Model. Comput. Simul. Volume 7, No. 2, April 1997, pp. 157-209.
21. S. Floyd, V. Jacobson, "Link-sharing and Resource Management Models for Packet Networks", ACM Transactions on Networking, Vol 3 No. 4, August 1995.
22. X. Xiao, L.M. Ni, "Internet QoS: A Big Picture", IEEE Network, Volume 13, N.2, March-April 1999, pp. 8-18.

A Policy Management Framework Using Traffic Engineering in DiffServ Networks

Elionildo da Silva Menezes, Djamel Fawzi Hadj Sadok, and Judith Kelner

Centro de Inform□tica
Universidade Federal de Pernambuco
50732-970 Recife PE Brasil
{esm,jamel,jk}@cin.ufpe.br

Abstract. The increasing growth and complexity of current corporate networks calls for higher level models and tools for managing network resources. In this sense, Service Level Management represents an important recent shift in system management. Furthermore, recently the IETF (Internet Engineering Task Force) DiffServ working group, has defined an architecture for implementing scalable service differentiation in the Internet, where network resources are allocated to traffic streams by service provisioning policies. This work presents and evaluates a functional architecture and a framework for mapping service management policies and constraints into DiffServ mechanisms. Various simulation scenarios described through the use of service policies are introduced and analyzed. It is shown that the use of service level management allows for efficient dynamic traffic engineering of DiffServ backbones.

Keywords: Quality of Service, DiffServ, Service Level Management, Network Management Policies

1 Introduction

The increasing growth of the Internet and corporate networks raises new concerns related to mechanisms such as routing, resource reservation, traffic engineering and management [7]. Furthermore, with the introduction of new Internet and corporate services such as e-business, VoIP (Voice over IP) and multimedia applications, the best-effort packet forwarding paradigm is not capable of attending the QoS (Quality of Service) requirements of such a wide range of services. This model deprives the network core of any form of intelligent traffic forwarding [22]. It is this design simplicity that contributed to the success of TCP/IP backbone technology.

This work presents both a model and an a functional architecture for the implementation and management of Internet services according to policies defined in the form of service contracts. Section two presents recent work onto management frameworks for QoS based management. Section three discusses both the model and a new architecture for contract based service management. Finally, section four presents various scenarios for contract based policy management of DiffServ domains whose simulation results are discussed in section five of this paper.

2 State of the Art

High level service management may be achieved through the definition and mapping of corporate policies onto network resources. Similarly, high level user contracts may be supported at the network level using mechanisms such as admission control, packet prioritization, traffic engineering, QoS routing and resource reservation. The IETF DiffServ working group has identified the need for service policies in order to allow border domain routers to correctly classify packets by consulting such policies. In this paper, a complete architecture shows how the integration between policy contracts and the DiffServ architecture is achieved.

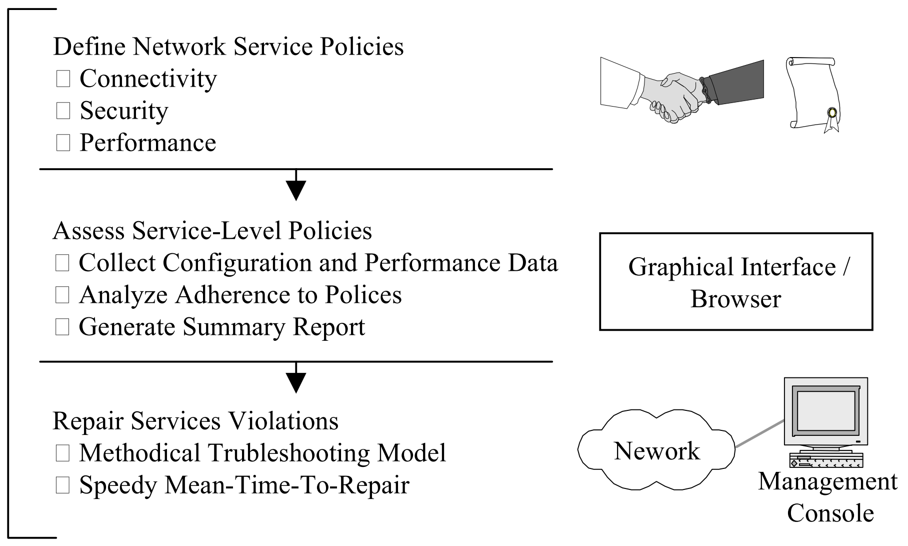

Define Network Service Policies
Connectivity
Security
Performance
Assess Service-Level Policies
Collect Configuration and Performance Data
Analyze Adherence to Polices
Generate Summary Report
Graphical Interface / Browser
Repair Services Violations
Methodical Trubleshooting Model
Speedy Mean-Time-To-Repair
Nework
Management Console

Fig. 1. Processes involved in Service Management

2.1 Service Level Management

SLM (Service Level Management) is a new paradigm for the integrated management of network resources, systems, applications and services according to policies defined with service contracts [19]. Such policies define rules and restrictions that control resource allocation to the supported services as shown in figure 1 [23] and expressed in the form of contracts or SLAs (Service Level Agreements). SLAs may determine the quality of offered services in terms of availability, security information, response time, delay, throughput, etc. SLAs may be static or dynamic. Static SLAs suffer little change limited to periodic review of the contracts between users and service providers. Dynamic SLAs, are designed to continuously and automatically adapt to network changes in order to maintain the service according to the contract.

SLM capable products include InfoVista□ [13], Netsys [23], HP IT Service Management [25] and Spectrum [6] although many of these implement limited SLM functionality.

2.2 Differentiated Services

Introducing QoS into what traditionally has been a best-effort packet network is not without its problems. Many IETF working groups have been setup in the last decade to address this issue. Among the adopted solutions, see [27]: the Integrated Services Model (IntServ) [3] which uses RSVP (Resource Reservation Protocol) [4] to make individual per flow reservations; the Differentiated Services Model (DiffServ) [2], a prioritization scheme for aggregations of flows; the MPLS (Multiprotocol Label Switching) [21] which uses establishes tag based forwarding paths through a network; the Traffic Engineering techniques [1] is also used to plan and configure network resources; and finally QoS based routing is used to compute routes that attend given QoS restrictions [9]. The QBone project [14] is currently piloting the use of the DiffServ packet prioritization behavior at core routers (known as Per Hop Behavior □ PHB). A clear advantage of DiffServ over the IntServ approach is scalability specifically when considering large backbones.

In the DiffServ architecture, border routers are responsible for aggregate classification of packets and their policing according to static or dynamic contracts or SLAs whereas core routers merely forward these marked packets according the DSCP (DiffServ Code Point) information within a packet, see figure 2 [17].

The combination of PHB based forwarding at the core and border packet classification and conditioning, allows a DiffServ domain to support various services. Note that the actual definition of services is outside the scope of the IETF and that so far two PHBs have been defined, namely, EF (Expedited Forwarding) [16] and AF (Assured Forwarding) [11]. The EF PHB offers rigid QoS guaranties and may be used to implement services such that require bound delay and guaranteed bandwidth. Examples of these include circuit emulation, voice and video services. The AF PHB on the other hand, offers limited QoS guarantees and may be used for applications such as Web access. The best-effort packet forwarding (BE) is maintained for low priority and background traffic.

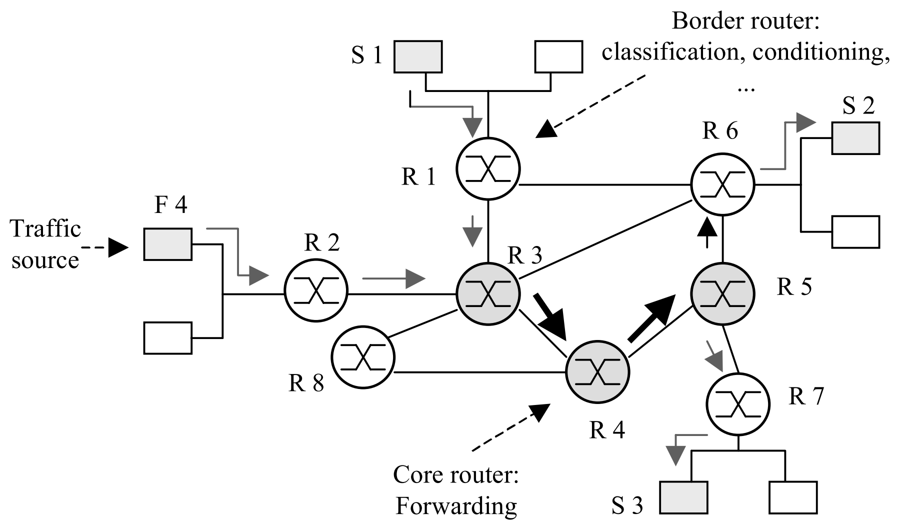


Fig. 2. Example of a DiffServ Network

2.3 Network Management Policies

Service policies are an important instrument in the correct implementation of QoS support since:

- SLAs are described in high abstract notations that are human readable;
- and are continuously consulted by network elements in order to assure that decisions at this level comply with corporate policies or user contracts.

Service policies are used in order to optimize, monitor and control the use of network resources and services. Policies may be based on parameters describing traffic origin and destination such IP addresses, port numbers, network and subnet information.

The IETF has been working on a common SLA specification syntax and semantics. The adopted notation defines a policy as consisting of a set of rules describing actions which should be undertaken under given situations [15]. Furthermore, more complex policies may be built combining simpler ones in order to ease their specification [24]. Many studies are underway to determine policy repositories and how these may be accessed [18]. Solutions vary from the adoption of existing systems and protocols including SNMP (Simple Network Management Protocol) [8], LDAP (Lightweight Directory Access Protocol) [12] and HTTP (HyperText Transfer Protocol) [10] to the creation of new ones such as in the case of the COPS (Common Open Police Service) [5]. It is likely that a combination of access techniques may be used.

3 A Service Management Model

The DiffServ IETF group has, according to its agenda, defined QoS mechanisms that heavily rely on the presence of policies to apply QoS related packet classification. On the other hand, work on SLM merely defines how services may be managed but lacks definition of the actual underlying mechanisms. The mapping of DiffServ policies into its actual mechanisms remains a challenge that is addressed in this section.

3.1 Proposed Model – Management Planes

The proposed model, shown in figure 3, structures service management into four distinct planes with increasing service abstraction levels. The fourth plane, the management plane, is orthogonal to the other three planes. These are described next:

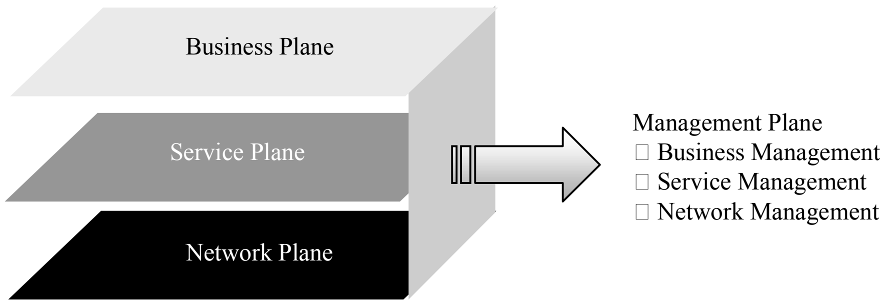


Fig. 3. A Four Planes Model for Service Management

– Business Plane: defines and evaluates services using human readable and less technical information in an effort to approximate the contract specification and management to the user. Users at the business level would be able to monitor their business services and act upon them without the need to deal with their implementation details. Business contracts may be drawn with items related to the time of offering the service, operation and maintenance costs, how fast services may recover, structure of support teams, relative service priority, penalties for contract violation and termination. Business contracts have judicial value and establish an agreement between clients and service provider. The latter is represented by the business manager. In this work, such contracts are referred to as CLAs (Customer Level Agreements).
– Service Plane: where services are defined using a more technical profile using QoS parameters such as delay, jitter, bandwidth, forwarding priority, traffic conditioning policy, redundancy schemes used to guaranty service availability, etc. Although service plane contracts are defined on an individual basis, they nevertheless involve the specification of requirements to be met by each of the DiffServ traffic aggregation classes. Here, a service manager is the entity responsible for the definition, monitoring and eventual service policy modification in order to attend service con-

tracts also known as SLAs (Service Level Agreements). A SLA may be seen as a set of CLAs of various clients as shown in figure 4.

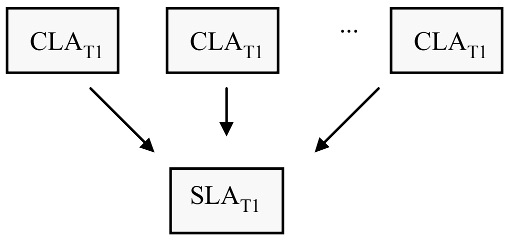


Fig. 4. SLA aggregating CLAs of the same type

– Network Plane: where are built the contracts that determine the technology used by the infrastructure in order to support the offered often commercial CLAs specified in the SLAs. At this plane, a contract is referred to as a TCA (Traffic Conditioning Agreement) in conformity with DiffServ terminology. TCAs, for example, can be used to establish parameters for router queues and routing algorithms on the basis of services. The network manager is the entity responsible for management duties at this plane.
– Management Plane: defines management activities to coordinate all three previous planes. It is important to emphasize the importance of the interaction between these planes in order to guaranty that contract changes are reflected at all levels and that business, service and network planes cooperate through the common management plane.

Although all planes include support for the fault, configuration, performance, accounting and security OSI management functional areas, the structure of management data may differ between the planes.

3.2 Functional Architecture

Figure 5 shows the architecture associated to the model described in the previous section. The depicted example is that of an Internet Service Provider (ISP) offering three DiffServ based commercial services, namely, Enterprise, Standard and Light. These are described next:

– ENTERPRISE: this service has the best performance when compared to the other two. It offers rigid bounded delay guaranties. Hence, it is ideal for delay sensitive applications such as videoconferencing. The network offers ENTERPRISE traffic priority over all other traffic classes. It is implemented using the DiffServ PHB EF and traffic conditioning is achieved through the discarding packets that are out of the negotiated profile. Studies have shown that provisioning EF based services, such the ENTERPRISE service, must not take up more than 5% of network resources to guaranty its correct operation.

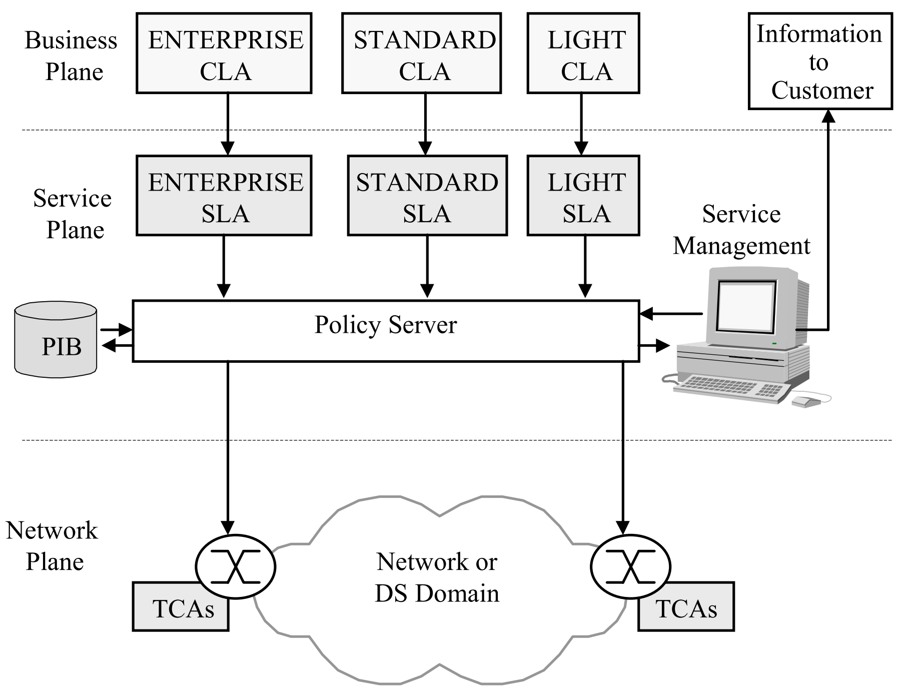


Fig. 5. A Functional Architecture for Service Management

– STANDARD: ideal for clients looking for a service that performs better than LIGHT, but cannot or would not pay for the usual limited and more expensive ENTREPIRSE service. This service offers minimum QoS guaranties, whereby the network seems lightly loaded. It is suitable for applications that are less sensitive to delay and bandwidth requirements such as Web navigation, file transfer and e-mail. This service is implemented using the DiffServ PHB AF, where traffic conditioning is achieved through the remarking of out of profile packets according to a negotiated service contract.
– LIGHT: characterized by its occupation of whatever is left of the network resources. This is, in other words, a best effort service with no real guaranties. It is expected that mostly background traffic for applications such as backups will use this type of service.

In the proposed architecture, CLAs associated to each of the three services (ENTERPRISE, STANDARD or LIGHT) are aggregated to form a corresponding SLA. These are then mapped into policies which are then stored into policy repositories, also known as Policy Information Bases (PIBs). PIBs are accessed by policy servers responsible for propagating the stored policies over network elements such as border routers, and configuring their TCAs in order to enable the border routers to perform traffic management within its DiffServ Domain. Periodically and in the case of important events, the policy server sends management reports to the service management application. This one in turn may collect this information in the form of user reports that it sends to them with information about whether their service contracts are being honored.

4 Simulations

In this section, a number of scenarios showing the use and mapping of service management concepts over DiffServ domains have been simulated. The objectives of such simulations include: validation the proposed architecture, showing its benefits, proving its viability, showing the important role of policy servers in controlling network performance and QoS. The simulations have been written using version two of the Berkeley network simulator [26] in both Tcl and C++ code. The network was configured to support Differentiated Services through the inclusion of DiffServ components for traffic conditioning, metering, shaping and packet dropping. Other functionality added to the simulator include support for EF, AF and BE PHBs.

4.1 The Simulated Network

Figure 6 shows the simulated network topology.

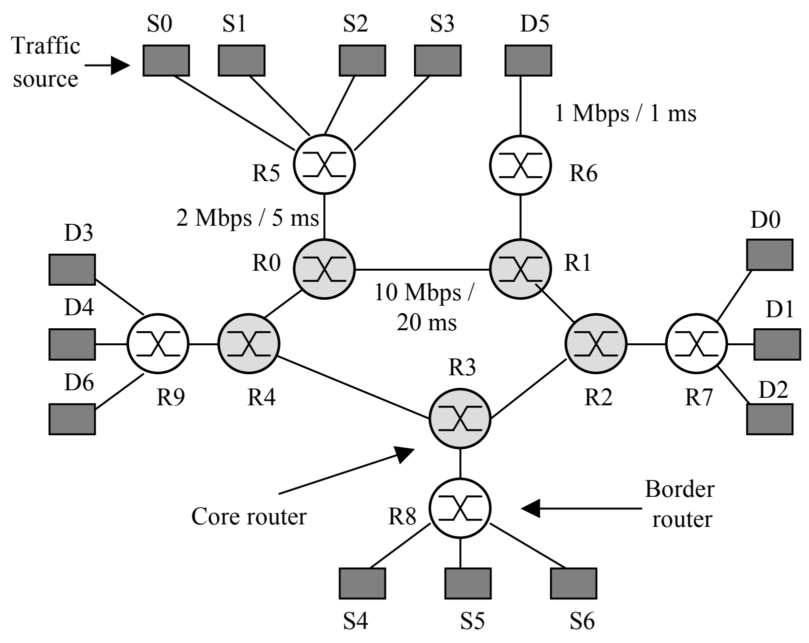


Fig. 6. Simulated Network Topology

4.2 Simulation Parameters

In order to approximate the simulated scenarios from real ones, a number of traffic sources have been defined to generate different traffic patterns. Table 1 shows a list of the sources used in the simulated network including CBR (Constant Bit Rate), alternate traffic fonts or On/Off, remote access using Telnet and file transfer using FTP.

Packet sizes have been configured to 576 bytes and 1 KB for CBR and other traffic sources respectively. Furthermore, all simulations are allowed to run for a minimum of 60 seconds. A single domain has been simulated in this work in order not to deal with inter-domain contracts which has been considered outside the scope of this work. The following network resource allocations have been made for the three DiffServ traffic classes: 5% for PHB EF, 40% for PHB AF and the remaining 55% have been allocated to the BE (Best-Effort) PHB.

In the first two case studies, each traffic shaper has been configured with the following parameters: peak rate of 500 Kbps, burst size of 16 KB and a queue length of 3. Whereas in the case of the third case study, the peak rate has been altered to 1 Mbps in order to use all the capacity of the links between sources S4 and S6 and the border router R8, which generate 1 Mbps each as shown in figure 6. Furthermore, at the scheduler used in all simulations, the parameters defined in [20] which also are presented in table 2.

Table 1. Parameters describing Traffic Sources used in the Simulation

Source	Traffic	PHB	Rate (Mbps)	Destination
S0	CBR	EF	0.1	D0
S1	TELNET	AF11	0.8	D1
S2	FTP	BE	-	D2
S3	FTP	BE	-	D3
S4	ON/OFF	AF11	1.0	D4
S5	CBR	EF	1.0	D5
S6	ON/OFF	AF11	*1.0*	D6

Table 2. Scheduler Configuration Parameters

ef-queue-length	40						
af-queue-length	62						
be-queue-length	150						
af-queue-rio-params	0.002	30	60	50	15	30	10
be-queue-red-params	0.002	50	145	20			
ef-queue-weght	1						
af-queue-weght	8						
be-queue-weght	11						
Aggregate-bytes-thresh	4000						

The complete code for these simulations and instructions of use may be found in http://www.di.ufpe.br/~servman/trabalhos.html.

5 Case Studies and Results

The simulated scenarios show the use of policy servers for service management and validate the architecture proposed in section 3.2. The terminology used to build the policy rules is based on definitions from [18].

5.1 First Case Study

A CLA representing a client contracting the ENTERPRISE service during periods of the morning and afternoon. It is assumed that the EF traffic from this service is associated to a CBR source with a 100 Kbps transmission rate, according to its SLA. In this case, it is desirable to configure the policy server so that it allows for this bandwidth allocated to this ENTREPRISE service to be used by other services and traffic classes at night for example. To achieve this, the policy used contains the following rules:

```
Rule 1: Offer HIGH priority for traffic from source S0,
between 7h and 19h
  if( (source == S0) && (timeOfDay == 0007-0019) )
  then
    priority = HIGH
  endif

Rule 2: Offer LOWER priority for traffic from source
S0, during day hours between 19h and 7h
  if( (source == S0) && (timeOfDay == 0019-0007) )
  then
    priority = 0
  endif
```

In order to simulate this scenario, the following parameters have been considered: source S0 is initially inactive during the first 30 seconds and stops at 60 seconds of simulation time. The graphs from figure 7, clearly show that when using the established policies, source S3 takes advantage of the available idle bandwidth during this period also showing how FTP traffic may be made to adjust to the available bandwidth. Since this is a best effort traffic source, only throughput has been measured. Other performance data such as delay and jitter are also considered in next two scenarios. Similar considerations are made about source S2. Furthermore, there was no change in S1's throughput, delay and jitter since its traffic follows the imposed SLA conditioning.

In other words, this scenario depicts the importance of actions from the policy server in the engineering and control of backbone traffic to ensure better use of network bandwidth and other resources.

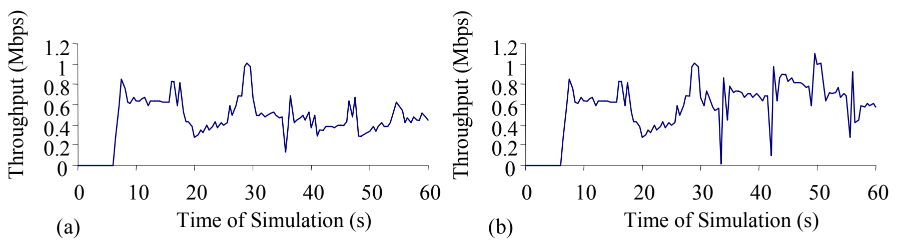


Fig. 7. S3 traffic source throughput. (a) Without the use of available bandwidth between 30s and 60s. (b) Using available bandwidth between 30s and 60s

5.2 Second Case Study

Next, a scenario where a user contracts the ENTERPRISE service for an application sensitive to both delay and jitter. The simulation assumes that at a given time, the default route used for forwarding packets from this application suffers a problem and becomes unavailable. A new route is then established although presenting higher delay and jitter value than those required by this application. The service policy for this CLA is described by next rule.

```
Rule 1: Monitor and notify service management about
contract violations
 if( (source == S0) && (LinkDown(R0,R1)) )
 then
   if( (delay_now>delay_S0) || (jitter_now>jitter_S0) )
   then
     LevelService = LOW
   endif
  endif
```

In the simulation source S0 generates traffic at a rate of 100 Kbps using the ENTERPRISE service. Figure 8, shows the actual throughput, delay and jitter associated to traffic flow. Although, both throughput and jitter have been maintained within the bounds of the service contract, the maximum delay has suffered a great deal. In the case of the application requiring delays inferior to 0.1 second, the policy server must alert both the policy manager to take action and the user to be aware of this contract violation.

5.3 Third Case Study

Our final case study considers the scenario of personal department making employees salary payments at five last days of each month for a 10 days duration. The payroll processing requires access to corporate databases distributed among two sites. Since the payroll activity for this company is considered as a very important service, a special service contract has been established between this department and its network

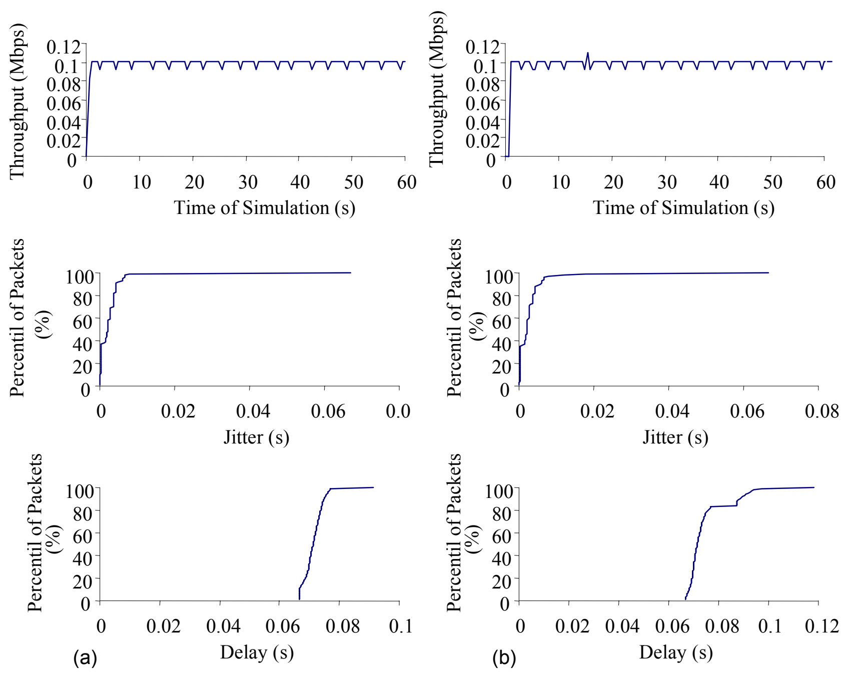


Fig. 8. Throughput, delay and jitter for S0. (a) Without link failure between R0 and R1 (b) With link failure between R0 and R1

provider which may be another company sector or a foreign entity. Note that this does not in anyway affect the architecture or the performance studies in this scenario. The contract stipulates that access during this period must be exclusive in terms of access to network resources. The STANDARD service has been selected for use in this scenario since the payroll application presents relatively flexible QoS parameters when consulting the databases. Such operations are generally sporadic and generate high bandwidth variable rate transmissions interleaved with periods of little activity.

The policies associated with this scenario are shown next:

```
Rule 1: Disable access to source S5 during this period
  if( (source == S5) && ((dayOfMonth in last5days) ||
(dayOfMonth in [1-5])) )
  then
    priority = 0
  endif

Rule 2: Offer exclusive access with maximum priority to
the department of personal during the period of payroll
  if( ((source == S4) || (source == S6)) && ((dayOf-
Month in last5days) || (dayOfMonth in [1-5])) )
  then
    priority = HIGH
  endif
```

```
Rule 3: Disable traffic generated by source 6 after
payroll has completed
  if( (source == S6) && (!(dayOfMonth in last5days) ||
!(dayOfMonth in [1-5])) )
  then
    priority = 0
  endif

Rule 4: Reset traffic priorities for sources 4 and 5 to
normal once payroll processing has completed
  if( ((source == S4) || (source == S5)) && (!(dayOf-
Month in last5days) || !(dayOfMonth in [1-5])) )
  then
    priority = NORMAL
  endif
```

In the simulation of this scenario, two On/Off traffic sources have been considered in order to attend high level requirements earlier specified. The time frame for elaborating the payroll is simulated as the time interval between 20 and 35 seconds. During this period, only sources S4 and S6 generate traffic at a 1 Mbps rate each. Source S5 is disabled and is only reactivated (allowed to transmit) after this period. Figure 9 shows the results of this simulation.

It is shown in this scenario that the policy server gives priority to both traffic flowing from both sources S4 and S6 while removing traffic from S5 during the payroll processing period. This procedure does not require the need for manual intervention. Outside the payroll time frame, default priority values are restored to all traffic sources. Note which this scenario is ideal to Corporate Intranet.

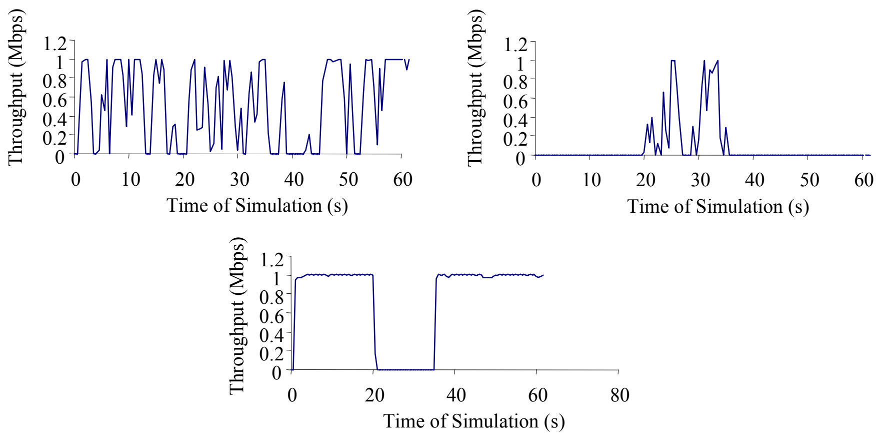


Fig. 9. Throughput for sources S4, S5 and S6. Above, throughput graphs for S4 and S6. Bellow, the throughput for source S5

6 Related Works

Some related works to the policy and contract management have appeared in the last years. However, none of them integrates all concepts with the DiffServ Architecture. In [28], contracts are created using only low-level notation based in DiffServ Architecture (queue length,jitter, PHBs, ...). In [29] and [30], layer-structured models are considered, but they arent integrated with DiffServ concepts. Thus, the main advantage of this paper is provides an approach which integrates Service Management and DiffServ Architecture concepts.

7 Conclusions

This work presented a four layered model for service management, based on DiffServ related policies. The mapping of policies between business, service and network layers has also been discussed. A functional architecture was described next through an example with various services with differing QoS requirements. The concepts of both the model and the associated architecture have been validated through the simulation of three scenarios. This work has shown the importance of integrating service level management through the use of policy servers in order to support different levels of QoS requirements. Using a DiffServ domain, it was shown that traffic engineering may be made through the implementation and management of service policies that prioritize flows, optimize traffic and monitor user contracts. It is important to state here that such control may be dynamic.

Although, the adopted scenarios are exclusive Internet backbone traffic, its is believed that this model and architecture apply to other scenarios such LANs using 802.1p.

Acknowledgements

Thanks to the referees for the good comments which helped to improve the final version.

References

1. Awduche, D., et al.: Requirements for Traffic Engineering over MPLS. draft-ietf-mpls-traffic-eng-01.txt. June 1999.
2. Black, D.: An Architecture for Differentiated Services. RFC2475. December 1998.
3. Braden, R., Clark, D. & Shenker, S.: Integrated Services in the Internet Architecture: An Overview. RFC 1633. June 1994.
4. Braden, R., et al.: Resource ReserVation Protocol (RSVP) Version 1 Functional Specification. RFC 2005. September 1997.

5. Boyle, J., et al.: The COPS (Common Open Police Service) Protocol. draft-ietf-rap-cops-06.txt. February 1999.
6. Lewis, Dr. Lundy: Spectrum Service Level Management Definition, Offerings, and Strategy. March 1998. http://www.cabletron.com/white-papers/spectrum/slm.pdf
7. Calvert, Kenneth L., Doar, M. B. & Zegura, E. W.: Modeling Internet Topology. IEEE Communications Magazine. June 1997.
8. Case, J. D., et al.: Simple Network Management Protocol (SNMP). RFC 1157. May 1990.
9. Crawley, E., et al.: A Framework for QoS-Based Routing in the Internet. RFC 2386. August 1998.
10. Fielding, E., et al.: HyperText Transfer Protocol HTTP/1.1. RFC 2616. June 1999.
11. Heinamen, J., et al.: Assured Forwarding PHB Group. RFC 2597. February 1999.
12. Howard, L.: An Approach for using LDAP as a Network Information Service. RFC 2307. March 1998.
13. InfoVista - Service Level Management Solution: Building a Service Management Environment. March 1998. http://www.3com.com/products/dsheets/400365a.html
14. Internet2, QBone Initiative. Available at http://www.internet2.edu/qos/qbone
15. Introduction to QoS Policies White Paper. July 1999. http://www.qosforum.com
16. Jacobson, V. & Nichols, K.: An Expedited Forwarding PHB. RFC 2598. February 1999.
17. Kurose, James F., and Ross, Keith W.: Computer Networking A Top-Down Approaching Featuring the Internet. http://www.seas.upenn.edu/~ross/book/Contents.htm
18. Mahon, H., Bernet, Y., and Herzop, S.: Requirements for a Policy Management System. draft-ietf-policy-req-01.txt. October 1999.
19. McConnell, John: Service Level Management Leveraging Your Network Investments. July 1998. http://www.3com.com/technology/tech_net/white_papers/500654.html
20. Murphy, Sean: Some notes on the use of my ns DiffServ Software. September 1999.
21. Rosen, et al.: Multiprotocol Label Switching Architecture. draft-ietf-mpls-arch-05.txt. April 1999.
22. Saltzer, et al.: End to End Arguments in System Design. ACM Transactions in Computer Systems. November 1984. http://www.red.com/Papers/EndtoEnd.html
23. Service Level Management with Netsys: A Model-Based Approach. July 1998. http://www.cisco.com/warp/public/cc/pd/nemnsw/nesvmn/tech/slnet_wp.htm
24. Strassner, J., et al.: Terminology for Describing Network Policy and Services. draft-ietf-policy-terms-01.txt. February 1999.
25. The HP IT Service Management Reference Model White Paper. March 1998. http://www.hp.com/pso/frames/services/whitepapers/wp-itsm-overview.html
26. VINT Network Simulator (version 2). http://www-mash-cs.berkeley.edu/ns
27. Xiao, X. & Ni, L. M.: Internet QoS: A Big Picture. IEEE Network. March/April 1999.
28. Gibbens, R. J., et al.: Na Approach to Service Level Agreements for IP Networks with Differentiated Services. Article submitted to Royal Society. 2000.
29. Pereira, P., and Pinto, P.: Algorithms and Contracts for Network and Systems Management. IEEE Latin American Networks Operations and Management Symposium. December 1999.
30. Wies, Ren: Policies in Networ and Systems Management Formal Definition and Architecture. Journal of Network and Systems Management. Plenum Publishing Corp. pp. 54-61. March 1994.

Quality of Service Issues in Multi-service Wireless Internet Links

George Xylomenos and George C. Polyzos

Department of Informatics
Athens University of Economics and Business
Athens, Greece 10434
`[xgeorge,polyzos]@aueb.gr`

Abstract. Internet application performance over wireless links is disappointing, due to wireless impairments and their adverse interactions with higher protocol layers. In order to effectively address these problems without the need for global protocol upgrades, we focus on link layer enhancement schemes. Simulations reveal that different schemes work best for different applications. We have thus developed a multi-service link layer architecture that can simultaneously enhance the performance of diverse applications by supporting multiple link mechanisms concurrently. Simulations confirm that this architecture provides dramatic performance improvements. The architecture can be embedded in various ways into the Internet. A critical issue for Quality of Service support over wireless links is the unpredictability of available resources. Our approach is based on fair sharing of the link before any measures are taken to improve the performance of individual traffic classes. This approach turns over the error control trade-off to the applications themselves.

1 Introduction

The Internet is always quick to adopt new communications technologies, largely due to the physical layer independent design of the *Internet Protocol* (IP), which offers a standardized interface to higher layers, regardless of the underlying link. The explosive growth of the Internet in the past few years is only paralleled by the growth of the wireless communications sector. *Cellular Telephony* (CT) is spreading worldwide and evolving towards 3G systems, while *Wireless Local Area Networks* (WLANs) are becoming inexpensive and conformant to international standards. Due to both physical and economic limitations however, wireless links consistently lag behind wired ones in performance.

The popularity of these wireless systems has generated considerable interest in their integration with the existing Internet. Even though satellites have long been a part of the Internet, most higher layer protocols and applications make assumptions about link performance that cannot be met by terrestrial CT and WLAN links. Thus, although providing IP services over wireless links is easy, their performance is disappointing. Since the Internet is evolving towards supporting *Quality of Service* (QoS) in order to enable new applications such as real-time multimedia communications, improving wireless link performance becomes even more critical.

This article presents an architecture that improves the performance of diverse Internet applications and extends QoS provision over wireless links. In Sect. 2 we outline the problem and review previous approaches, arguing for a link layer solution. Our simulations show that different applications favor different enhancement schemes. In Sect. 3 we present a *multi-service link layer* architecture that simultaneously enhances the performance of diverse applications by supporting multiple link mechanisms in parallel, without any changes to the rest of the Internet. The remainder of the article discusses how our architecture fits into the context of Internet QoS approaches: Sect. 4 covers the existing best-effort service, Sect. 5 the Differentiated Services architecture, and Sect. 6 a dynamic service discovery architecture.

2 Background

IP provides unreliable packet delivery between any two network hosts: packets may be lost, reordered or duplicated. Applications can use the *User Datagram Protocol* (UDP) for direct access to this service. Some UDP applications assume that the network is reliable enough, for example file sharing via the Network File System over LANs. Delay sensitive applications may also use UDP, adding their own custom error recovery mechanisms. For example, real-time conferencing applications may introduce redundancy in their data to tolerate some errors without the need for slow end-to-end retransmissions.

The majority of applications however require complete reliability. Those usually employ the *Transmission Control Protocol* (TCP), which provides a reliable byte stream service. TCP breaks the application data into segments which are reassembled by the receiver. The receiver generates cumulative acknowledgments (ACKs) for those segments received in sequence, with duplicate acknowledgments (DUPACKs) for reordered ones. Although a lost data segment leads to a DUPACK when the next segment arrives, since packets may be reordered by IP, only after multiple (usually 3) DUPACKs are returned does the sender retransmit the (apparently lost) next segment in sequence. The sender also tracks the round trip delay of the connection, so that if a segment is not acknowledged on time it is retransmitted on an end-to-end basis [1].

Since data corruption is extremely rare in wired links, TCP assumes that all losses are due to congestion. Thus, after a loss the sender reduces its transmission rate to allow router queues to drain, and then gradually increases it so as to gently probe the network [1]. Wireless links are considerably less reliable than wired ones though. This causes UDP applications to fail when used over wireless links with worse quality than expected. TCP applications on the other hand repeatedly reduce their transmission rate when faced with frequent wireless errors, so as to avoid what is (falsely) assumed to be congestion, thus dramatically reducing their throughput. Longer paths suffer more, since end-to-end retransmissions increase delay, a critical issue for interactive applications.

We have focused on CT and WLAN systems which are widely available and relatively inexpensive. The low (terrestrial) propagation delays of such systems differentiate them from traditional (geostationary) satellites. Present CT systems support low bit rates in the wide area while WLAN systems support higher speeds and lower error rates. *Personal Communications Systems* (PCS), the evolution of CT, will provide higher bit rates using smaller coverage cells. WLANs suffer from losses of up to 1.5% for Ethernet size packets [2], while CT systems suffer from losses of 1-2% for their much shorter (voice oriented) frames [3]. The effects of such losses are dramatic: a 2% packet loss rate over a single WLAN link reduces TCP throughput by half [4].

The performance of UDP applications over wireless links has largely been ignored, mainly due to the diversity of UDP applications. Considerable work has been devoted to TCP however, as it is the most popular Internet transport protocol. The goal is to avoid triggering end-to-end congestion recovery due to wireless errors. One way to achieve this is to split TCP connections into one connection over the wireless link and another one over the remainder (wired part) of the path, bridged by an agent [5]. Recovery from wireless errors is performed locally over the wireless link. Although this speeds up recovery, it violates transport layer semantics. In addition, TCP error recovery is inefficient and unable to take advantage of link specific optimizations. Split TCP connections are also incompatible with IP security which encrypts TCP headers [6].

The main alternative to transport layer solutions is local error recovery at the link layer. The Radio Link Protocols (RLPs) provided by CT systems offer error control customized for the underlying link [3]. Since they always apply the same scheme though, they may not be appropriate for some traffic. For example, retransmissions may delay real-time traffic or interfere with TCP recovery [7]. One way to avoid such adverse interactions is to exploit transport layer information at the link layer. By *snooping* inside TCP segments we can transparently retransmit lost segments when DUPACKs are returned, hiding the DUPACKs from the sender [4]. This scheme avoids link layer error control overhead and outperforms split TCP schemes [4]. However, it works only in the direction from the wired Internet towards the wireless host due to its reliance on TCPDUPACKs (in the reverse direction, DUPACKs are returned too late for local error recovery) [8] and it is also incompatible with IP security.

Despite their limitations, link layer schemes are preferable to higher layer ones because they provide a local solution to a local problem. Therefore, they can be customized for the underlying link and deployed transparently to the rest of the Internet. In order to examine the performance of various link layer enhancement schemes in conjunction with a diverse set of applications, over a range of wireless error models, we performed extensive simulations using the Network Simulator version 2 (ns-2) [9]. We thus studied a wide range of parameters under controlled conditions. The details of the simulation set-up, the experiments and the results are presented elsewhere [10]. Here we only provide a small sample of our results to motivate the discussion on Quality of Service issues.

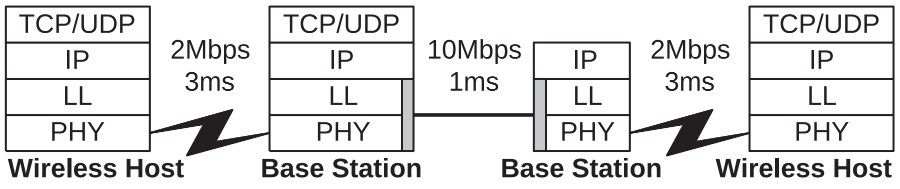


Fig. 1. Simulation topology

We simulated two topologies, both depicted in Fig. 1. In the first scenario two Wireless Hosts communicate over two identical but independent wireless links and a single LAN link with 10 Mbps speed and 1 ms delay.[1] In the simpler second scenario, a Wireless Host communicates with a Base Station over a single LAN and a single wireless link. Unlike the former scenario which is symmetric, in the latter one the Base Station is the "server" or "sender," i.e. most data flows from the Base Station towards the Wireless Host. However data may also flow in the reverse direction, for example TCP ACKs. The wireless links are WLANs with 2 Mbps speed and 3 ms delay, using 1000 byte packets. They suffer from bit errors which occur at exponentially distributed intervals with average durations between 2^{14} and 2^{17} bits [4], which lead to packet loss rates of 0.8% to 5.9%.[2]

We tested both TCP and UDP applications so as to examine the benefits of various link layer schemes. For TCP, we simulated the *File Transfer Protocol* (FTP), measuring the application level throughput of a 100 MByte transfer. We used the TCP Reno implementation of ns-2 with 500 ms granularity timers. For UDP, we simulated real-time conferencing with a single sender (speaker) and a single receiver. We used a speech model where the speaker alternates between talking and silent states with exponential durations averaging 1 s and 1.35 s, respectively [11]. When talking, the speaker sends audio and video at a *Constant Bit Rate* (CBR) of 1 Mbps. We measured the application level packet loss rate and a delay metric consisting of mean packet delay plus twice its standard deviation, to account for the effects of variable delays, over a 500 s interval.

For TCP, in addition to the *Raw Link* (native) service, we studied a full recovery *Selective Repeat* (SR) scheme which allows multiple negative acknowledgments (NACKs) per loss [12], and *Karn's RLP*, a limited recovery scheme which gives up on losses that persist for a number (3 for TCP tests) of retransmissions [3]. We also tested the *Berkeley Snoop* scheme which employs transport layer information for link layer recovery [4]. Our FTP tests show that the TCP unaware link layer schemes (SR and Karn's RLP) improve throughput (compared to Raw Link) by 15-2900%, depending on the topology and native loss

[1] We also simulated a WAN path, which is abstracted as a slower, higher delay link.
[2] We also simulated slower CT and PCS links, with completely different loss models.

rate. Berkeley Snoop however fails in the two wireless link scenarios, as it is unable to retransmit from the Wireless Host towards the Base Station.[3]

For UDP, we tested schemes that provided limited recovery in exchange for lower delays. In addition to Karn's RLP, we designed an *Out of Sequence* (OOS) RLP, which releases packets to higher layers as they arrive, and not in sequence (as in SR and Karn's RLP). We also studied a block based *Forward Error Correction* (FEC) scheme with 1 parity for every 12 data packets [10]. Our real-time conferencing tests show that both RLP schemes dramatically reduce losses, even with 1 retransmission per loss, unlike the FEC scheme whose gains do not justify its overhead. The OOS RLP variant significantly reduces the delay of Karn's RLP, with gains that become more attractive with multiple wireless links. While TCP favors in sequence delivery, OOS RLP is perfectly adequate for real-time applications with their own resequencing buffers. For the low delay WLAN links, OOS RLP actually leads to lower delays than the FEC scheme studied.

3 Multi-service Link Layer Architecture

The results presented briefly above (see [10] for additional details) show that link layer error recovery dramatically improves Internet application performance over wireless links. Link layer solutions can be locally deployed and customized for the underlying link, without any changes to the rest of the Internet. Our results also show however that different schemes work best for the TCP and UDP applications tested, and it is very likely that other applications will favor quite different schemes. Therefore, what we need at the link layer is a flexible multi-protocol approach. We have developed a *multi-service link layer* architecture, which provides multiple link enhancement services in parallel over a single physical link. Each service fits the requirements of a generic class of applications, such as TCP based or real-time UDP based. To simplify service design, the architecture assigns incoming packets to services and fairly shares the available bandwidth. As a result, services are isolated and unaware of each other, which allows them to be easily added and removed as new needs arise.

We summarize below the design of our architecture, while the following sections focus on its interface with various Internet Quality of Service approaches.[4] Figure 2 gives an outline of our scheme. Incoming packets are classified and passed to the most appropriate service, based on their application. A simple classifier may use the IP protocol field to distinguish between TCP and UDP and the TCP/UDP port field to determine the application in use. Alternatively, when the *Differentiated Services* (DS) architecture is used for QoS provision at higher layers, the classifier may exploit the IP DS field [14], which is visible even with IP security. Packets that cannot be matched to any enhanced service are mapped to the default, best-effort, service. Each service operates in isolation, using retransmissions, FEC, or any other mechanism desired, and keeping its own buffers and timers, that may be optimized for the underlying link. Outgoing

[3] This limitation is even more apparent with (bi-directional) interactive applications.
[4] The multi-service link layer architecture was briefly introduced in [13].

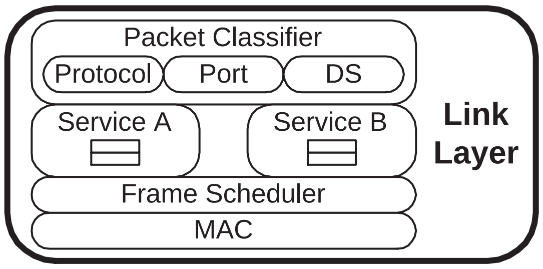


Fig. 2. Multi-service Link Layer Architecture

frames are passed to a scheduler which tags each frame with a service number and eventually passes it to the MAC sublayer for transmission. At the receiver, frames are passed to the appropriate service (based on their tags), which may eventually release them to higher layers.

Since each service is free to arbitrarily inflate its data with error recovery overhead, we must use a frame scheduler to prevent heavily inflated data streams from monopolizing the physical link. We chose a *Self Clocked Fair Queueing* (SCFQ) scheduler [15] which can strictly enforce the desired bandwidth allocation for each service when the link is loaded. When some services are idle, their bandwidth is proportionately shared among the rest. Figure 3 gives an outline of the scheduler. The rate table shows the fraction of the link allocated to each service. We can set these rates statically, or the classifier may dynamically set them to equal the fraction of incoming traffic that it allocates to each service, before error recovery takes place. In this case, the best-effort service will receive exactly the same bandwidth as without any link layer enhancements.

Frames awaiting transmission are buffered in service specific queues. The scheduler maintains a virtual time variable which is equal to the time stamp of the last packet transmitted. To determine the time stamp of an incoming packet, we divide its size by its service rate, and add it to the time stamp of the previous frame in its queue. If its queue is empty, we use the current virtual time instead. When the link is freed, the frame with the lowest virtual time is dequeued, the system virtual time is updated, and the frame starts transmission. The scheduler organizes all service queues in a heap sorted by the time stamp of their first frame, so that the next frame to transmit is always at the top. The heap is re-sorted when a frame is dequeued for transmission or when an empty queue receives a new frame. Thus, each frame requires a simple calculation for its time stamp and $\log_2 n$ operations (for n services) to re-sort the heap when the frame leaves (and, possibly, when it enters) the scheduler.

Our multi-service link layer architecture can be locally deployed, transparently to the rest of the Internet. Additional services can be provided by inserting new modules and extending the mappings of the packet classifier, which may be reused over any wireless link. Services may be optimized for the underlying link,

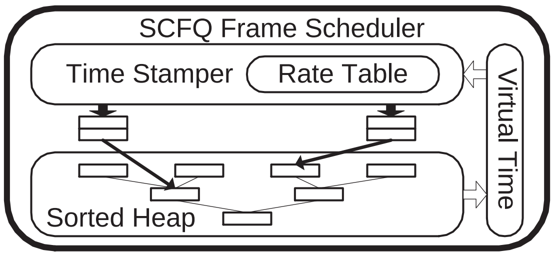


Fig. 3. Self Clocked Fair Queueing frame scheduler

freely selecting the most appropriate mechanisms for their goals. The scheduler ensures that services fairly share the link despite their variable overheads. Service rates may be set statically or dynamically, while applications may be mapped to services either heuristically, or, in the presence of higher layer QoS schemes, intelligently. To verify our claims that the multi-service link layer architecture can simultaneously enhance the performance of diverse applications, we repeated our simulations with the TCP and UDP applications executing in parallel [10]. The multi-service link layer approach provided similar gains to those in single application tests. FTP performance improved by 11-2425%, depending on the topology and native loss rate, while conferencing delay was kept low since the scheduler prevented the TCP data stream from stealing UDP bandwidth.

4 Best-Effort Service Interface

A critical component of a multi-service link layer is a function for mapping IP packets to available services. The lowest common protocol layer on the Internet, the network layer, provides only a single, best-effort, packet delivery service. The assumption is that higher layer protocols can extend this service to satisfy additional application requirements. Our simulations show however that multiple link layer services are needed to enhance Internet application performance over wireless links. Since IP and the protocols using it are not aware of multi-service links though, they cannot map their requirements to available services.

In order to remain compatible with the existing Internet infrastructure, our architecture should perform this mapping without changing the interface between the link and network layers. Due to the scale of the Internet, such changes would take years to propagate everywhere. Performance should always be at least as good as with a single service, i.e. enhancing the performance of some applications should not degrade the performance of others. Finally, applications should never be mapped to services that degrade their performance. These requirements ease deployment as they allow services to be introduced gradually, to selectively enhance performance for some traffic, without affecting the rest.

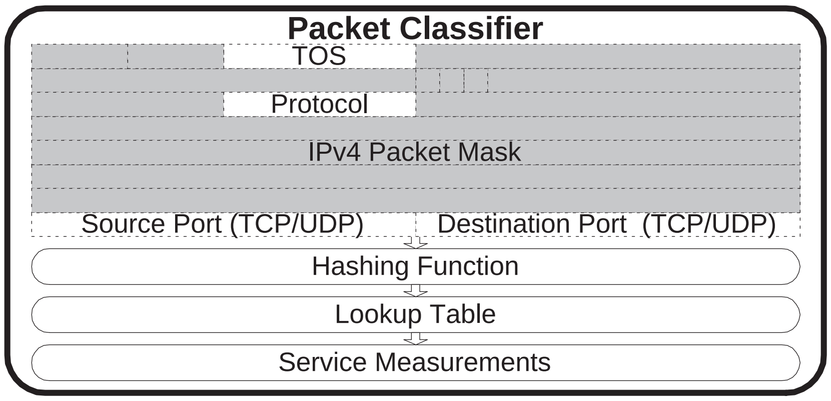


Fig. 4. Heuristic packet classifier

Our link layer has single entry and exit points to ease its integration with existing protocol stacks. Since we cannot change this interface, the only information we can use for service selection is the incoming IP packets themselves. In order to match application requirements with available services we can use a classifier which employs heuristic rules to recognize certain applications based on IP, TCP and UDP header fields. All other traffic uses the default (native) link service. Figure 4 shows data flow in this heuristic classifier for IPv4. IP packet headers are masked to isolate the fields needed for classification (masked fields are grayed out). These fields pass through a hashing function that produces an index to a lookup table, whose entries point at the available services. Unknown applications are mapped to entries pointing at the default service, which provides the same performance as with a single service link layer.

Whether higher layers perform packet scheduling or not, they expect the link layer to allocate link bandwidth as if only a single service was available. For classification to be transparent, the services sharing the link should respect this (implicit) bandwidth allocation, even though each service may add arbitrary amounts of recovery overhead. In our architecture, after packets are assigned to services the classifier uses their size to implicitly deduce the share of the link allocated to each service. Over an implementation dependent interval, the classifier divides the amount of data assigned to each service by the total amount of data seen to get a fraction r_i, where $\Sigma_{i=1}^{n} r_i = 1$, for each of the n services. These fractions are used in the rate table employed by the frame scheduler. Thus, unknown applications are allocated at least the same amount of bandwidth as in a single service scheme, while known applications trade-off throughput, when the link is loaded, for better error recovery. The header mask, hashing function and lookup table are provided by an external administrative module that is aware of application requirements, header fields and service properties.

Heuristic packet classification is based on the Protocol field of the IP header, which indicates TCP, UDP, or another protocol. All TCP applications can be mapped to a single TCP enhancement service, implemented by one of the link layer schemes mentioned above. UDP applications have very diverse requirements though, so decisions must be made on a per application basis. Known UDP applications can be recognized by looking at the source and destination port fields of the UDP header. Many applications use well known ports to communicate, for example to allow servers to be located at remote hosts. Thus, the protocol field along with well known ports can be used to recognize applications and map their data to the most appropriate services. Another field that may be used is the *Type of Service* (TOS) field of the IPv4 header. Originally, four bits were defined to indicate preference for reduced delay, reduced cost, increased throughput and increased reliability, while three more bits were defined to indicate packet priorities. The TOS field, however, is rarely used and it is being redefined to support Differentiated Services. The same holds for IPv6, where the *Traffic Class* field is also being redefined to support Differentiated Services.

A significant drawback of heuristic classifiers is the amount of effort required to construct them. Applications and services must be manually matched for each type of link whenever new applications or services are added. Many applications will not be recognized, not only due to the large amount of applications in existence and the constant appearance of new ones, but also because many applications do not use well known ports at all. Although some QoS provisioning approaches combine the transport protocol, source/destination port and host address fields to classify packets [16], maintaining state for all these combinations requires end-to-end signaling and considerable storage, both of which are not available at the link layer. A more important problem for heuristic classifiers is that IP security mechanisms encrypt the source/destination ports of TCP and UDP headers and replace the value of the protocol field with the identifier of the IP security protocol [6], leaving only the TOS field visible, which is currently inadequate to describe application requirements.

5 Differentiated Services Interface

The single best-effort service offered by IP is reaching its limits as real-time applications migrate from the circuit switched telephone network with its explicit delay guarantees to the packet switched Internet. For these applications to exploit the reduced costs offered by statistical multiplexing, some type of performance guarantees must be introduced on the Internet. Users would presumably pay more for better Internet service if it would be cheaper to use the Internet rather than a circuit switched network for the same task. The main problem with Internet QoS is generally considered to be congestion. Applications may not be able to get the throughput they need due to contention for limited link resources and their end-to-end delay may increase due to queueing delays at congested routers. In addition, when router queues overflow, data loss occurs. UDP applications have to deal with these losses themselves, while TCP applications face additional delays due to end-to-end TCP recovery.

One approach for Internet QoS provision is the *Integrated Services* architecture [17], which has been criticized in two ways. First, it must be deployed over large parts of the Internet to be useful, since its guarantees rely on actions at every router on a path. Second, it requires resource reservations on a per flow basis, where a flow is defined as a stream of data between two user processes with the same QoS requirements. Since a huge number of flows exists, the scalability of any QoS scheme based on per flow state is limited. In IPv6 a 20 bit flow label is defined in the IP header to identify flows between two hosts, while host addresses are expanded to 128 bits. The only solution to this problem is aggregation of flow state, but it is unclear how this can be achieved.

An alternative approach that aims to avoid these limitations is the *Differentiated Services* architecture [14]. In this scheme flows are aggregated into a few classes, either when entering the network, or when crossing network domains. At these points only, flows may be rate limited, shaped or marked to conform to specific traffic profiles, which are either negotiated between users and network providers or between neighboring domains. Within a domain, routers only need to select a *Per-Hop Behavior* (PHB) for each packet, based on its class. This is denoted by the 8-bit Differentiated Services (DS) field of the IP header, which subsumes both the IPv4 TOS and the IPv6 Traffic Class field. State aggregation into a few classes means that this scheme scales well, but the guarantees that may be provided are not as fine grained as with Integrated Services.

This architecture intentionally leaves the definition of PHBs open, to allow experimentation with different schemes. As an example, the *expedited forwarding* PHB provides a minimum amount of bandwidth at each router, for traffic that is rate limited when entering the network or the domain so as not to exceed this bandwidth. This PHB provides low delay and loss by eliminating congestion for a class, offering a virtual leased line service. As another example, the *assured forwarding* PHB group defines a number of service classes, with each one allocated a specific share of the bandwidth. Within each class packets may have multiple levels of drop preference. Besides scheduling so as to satisfy the bandwidth requirements of each class, when a class is congested routers should drop first the packets with highest drop preference. Flows are marked with higher drop preference levels when they exceed the traffic profile for their class, rather than being rate limited. Both PHBs may be implemented by many scheduling mechanisms and queue management schemes.

The services provided by this architecture depend on the PHBs available and are meant to provide generic QoS levels, not application specific guarantees, hence the mapping of traffic classes instead of flows to PHBs. Only network entry points must be aware of both application requirements and PHB semantics to perform flow aggregation. Similarly, only domain entry points must be aware of the semantics of PHBs available in their neighboring domains to perform appropriate translations. Traffic policing, shaping and marking, is also only performed at these points. For neighboring domains these profiles should be relatively static as they represent large traffic aggregates, while at network entry points they could be frequently modified by the user. This is far more economical than the Integrated Services approach with its flow specific signaling.

Differentiated Services and multi-service link layers are solutions to orthogonal but complementary problems. Differentiated Services are concerned with congestion and its impact on throughput, delay and loss. The services provided are based on link independent PHBs, supported by packet scheduling and queue management mechanisms at the IP level. Multi-service link layers are concerned with recovery from link errors customized to each type of application. The error recovery mechanisms used are link dependent and local, thus they cannot be standardized into a common set of link independent PHBs. The frame scheduling provided only protects services from each other by mirroring higher layer allocations, it does not provide end-to-end guarantees. By only providing Differentiated Services over wireless links we may offer to applications a nominal IP level QoS, but their actual performance will be limited by link losses. For example, even if we reserve wireless link bandwidth for a TCP traffic class, only a small fraction of it will be used due to losses and inefficient TCP error recovery. Multi-service link layers provide adequate recovery to fully utilize wireless links, but they also need higher layer guidance to perform scheduling.

These two architectures are excellent complements to each other. Differentiated Services provide congestion control, using packet scheduling and queue management, while multi-service link layers add application dependent error control, respecting higher layer scheduling despite the introduction of recovery overhead. They both offer a few services at each node (PHBs or link layer schemes) for aggregated traffic classes with common requirements. They can be combined by extending the DS field to also specify the error requirements of each traffic class. For example, a traffic class with a reserved amount of bandwidth could be subdivided into two subclasses with different error recovery requirements by using one bit of the DS field. Each subclass would be mapped to an appropriate link layer service. All DS bits are only set when flows are aggregated into classes. Applications could indicate their requirements when injecting their traffic into the network, with boundary routers translating them to local equivalents when required. The result is a simplified multi-service classifier, shown in Fig. 5 for IPv6 packets. The DS field is isolated via a header mask, and then a hashing function plus a lookup table map it to an appropriate service. The same procedure can be used with IPv4 headers, where the DS field is the original TOS field, instead of the Traffic Class field of IPv6 headers.

This classifier does not rely on multiple header fields and complex rules to determine application requirements. A single field is used with well defined semantics. New applications may be mapped to existing classes if their requirements are similar to those of previous applications. More importantly, the DS field is not hidden by IP security mechanisms, it is visible even in encrypted packets. Since the Differentiated Services module performs scheduling and queue management, traffic entering the multi-service link layer already obeys the required bandwidth allocations. For example, two TCP traffic subclasses belonging to

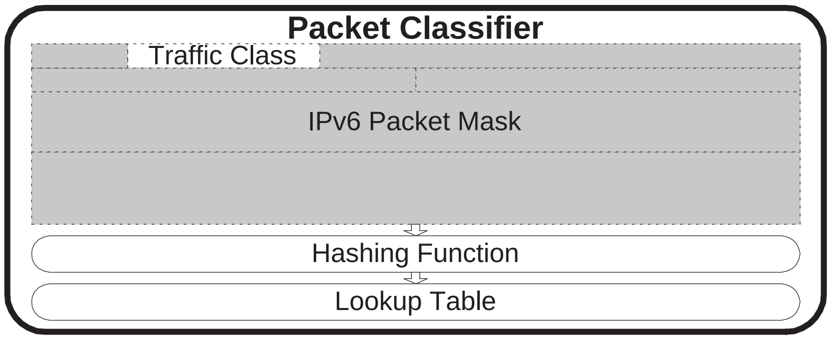


Fig. 5. Differentiated Services packet classifier

separate classes may be rate limited in different ways at the IP level. At the link layer, however, they are already shaped as needed, hence they can share the same service and frame scheduler queue without introducing congestion. Thus, regardless of the number of traffic classes, the multi-service link layer must only maintain a single instance of each service. The service rates can be set in two ways. If the subclasses of each traffic class have separate bandwidth allocations at the IP level, then the bandwidth for all subclasses mapped to the same service is added to set its service rate. If subclasses share a common bandwidth pool within each class, then a service measurements module is inserted after the lookup table, as in Fig. 4, to automatically determine service rates.

6 Advanced Quality of Service Interface

The performance of the services available at each wireless link can vary widely. Different links may favor different error recovery schemes, with the performance of each scheme varying over time due to environmental conditions. If a characterization of the end-to-end performance of a network path is provided, applications can verify that a given service is suitable for their needs [18]. In addition, when path characteristics change significantly, fresh characterizations enable adaptive higher layers to modify their policies. In a hierarchical cellular system,[5] horizontal handoffs may cause error behavior to change, while vertical handoffs alter all wireless link characteristics. To describe the services offered over a network path, we must dynamically discover what is provided at each link. This can be achieved if each service dynamically characterizes its performance with a set of standardized metrics. Dynamic characterization means that performance may

[5] These are composed of multiple overlaid cellular systems: satellites cover the whole globe, cellular systems cover populated areas, and indoor WLANs cover buildings. Thus, in addition to horizontal handoffs between adjacent cells of the same system, the user can perform vertical handoffs between different systems.

be evaluated as often as needed, while metric standardization means that higher layers will be able to assess the performance of arbitrary services without any knowledge of the link layer mechanisms employed. An end-to-end QoS module will thus be able to compose link metrics into end-to-end path metrics.

We have defined three link independent metrics reflecting the possible trade-offs in each error recovery scheme: goodput, loss and delay. Metrics are calculated dynamically over an implementation dependent interval. Reported metrics may be smoothed by using a weighted average of the latest calculated value and the previous reported value, as with TCP round trip delay estimates. *Goodput* (g_i, for service i) is the ratio of higher layer data transmitted during the measurement interval to link layer data transmitted, including all overhead. The amount of higher layer data transmitted may differ from the amount received due to residual losses. Goodput can be calculated at the sender without receiver feedback. *Loss* (l_i), is the ratio of higher layer data lost to higher layer data transmitted (lost plus received). Loss is calculated by the receiver based on the sequence of data released to higher layers. It depicts the residual loss rate after error recovery. It is greater than zero for limited recovery schemes. *Delay* (d_i) is the one way average delay (in seconds) for higher layer packets. Delay can be estimated at the receiver based on knowledge of the implemented scheme and wireless link characteristics. Retransmission schemes could add one round trip delay estimate for each retransmission to their one way delay estimate. A FEC scheme could add the interval between loss of a recovered frame and its reconstruction from parity data to its one way delay estimate.

Delay denotes the error recovery delay of a service, since there is no congestion inside the link layer. It can be added to IP level queueing delay to give the total delay for each node, which can be used to estimate end-to-end delays incorporating both congestion control and wireless error recovery. Delay sensitive traffic subclasses can choose the lowest delay service whose residual loss falls within their tolerance limits. Goodput can be combined with loss to get *Effective Goodput* (e_i), defined as $e_i = g_i{*}(1-l_i)$, or, the ratio of higher layer data received to link layer data transmitted. Essentially, e_i shows how much of the bandwidth allocated to a service is used for data actually received, after discounting error recovery overhead and residual losses. If the link bandwidth is B and service i is allocated a service rate r_i in the scheduler, then its throughput is $B * r_i * e_i$. This may be used to estimate the throughput for each service given a set of service rates, or to calculate the service rate needed to achieve a target throughput for a particular service. The reason for using goodput instead of throughput for service characterization is exactly that goodput, unlike throughput, can be used to predict service behavior with different rate allocations.

These metrics may be used at multiple layers to serve different needs, as shown in Fig. 6. The physical layer provides hardware dependent information, such as fixed one way delay, that may be used by the link layer services to provide their link independent g_i, l_i and d_i metrics. At the network layer, scheduling mechanisms such as CBQ may use those metrics to set bandwidth allocations for each service. End-to-end QoS schemes may use RSVP to gather information

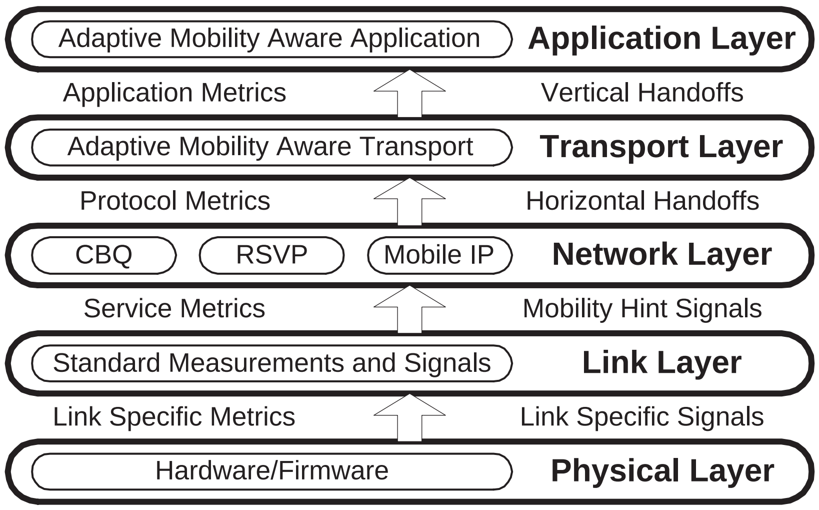


Fig. 6. Propagation of service measurement and mobility feedback

about node services in order to estimate end-to-end characteristics. Path characteristics can be used by both transport protocols and applications to adapt their operation to prevailing conditions. For example, TCP may limit its congestion window to respect available bandwidth limitations, or, video conferencing applications may select encoding schemes that can deal with the residual loss rate of the path. Service metrics can be refined at each layer, as in the network layer where local metrics are used to compose end-to-end metrics.

To help higher layers deal with mobility, we can extend this interface to provide mobility hints to interested parties via upcalls, as shown in Fig. 6. The link layer can use hardware signals and combine them with its own state to flag events such as connections and disconnections. If a handoff is taking place, higher layers will receive one disconnection and one connection upcall from different links. These link independent upcalls can be used by the IP mobility extensions to allow fast detection of handoffs, instead of relying on periodic network layer probes [19]. Higher layers may be notified by the network layer of horizontal and vertical handoffs via further upcalls. TCP may be notified of pending horizontal handoffs to temporarily freeze its timers and avoid timeouts during disconnection intervals. A video conferencing application may be notified of vertical handoffs so as to change the encoding scheme used to a higher or lower resolution one, depending on the available bandwidth. The characteristics of the new path can be discovered by using end-to-end QoS provisioning mechanisms to query the metrics exported by the new wireless link.

This QoS interface was designed to fit the needs of *One Pass With Advertising* (OPWA) [18] resource reservation mechanisms. One pass schemes cannot specify a desired service in advance as they do not know what is available on a path [16], thus the resources reserved may provide inadequate service. Two pass schemes specify a service in advance but make very restrictive reservations on the first pass, relaxing them on a second pass [20]. Such reservations may fail due to tight restrictions on the first pass. In the OPWA scheme, an advertising pass is first made to discover the services available on the path, and then a reservation pass actually reserves the resources needed for the selected service. Our interface allows OPWA schemes to discover throughput, delay and loss restrictions imposed by wireless links, by looking at local multi-service metrics. After that information is gathered, applications may choose a service, and the reservation pass can set up appropriate state so as to provide it. Mobility hints notify higher layers that they should revise their path characterizations after handoffs, in order to satisfy the needs of adaptive mobility aware applications that can adapt their operation based on resource availability.

A related proposal was made in the context of IP mobility: a router could notify hosts communicating with a wireless host of new link characteristics after a handoff [21]. This approach however does not support link characterization between handoffs or multiple services. Another related proposal is an adaptation interface for mobility aware applications that notifies applications when their available bandwidth deviates from a specified range [22]. An application supporting multiple encodings of its data stream selects a bandwidth range for each, and switches encodings whenever the available bandwidth moves into another range. This interface is not appropriate for the link layer, as it requires end-to-end signaling and per application state, but it can be easily implemented on top of our mobility hints. A QoS management module at each host would accept bandwidth range requests from local applications. Instead of constantly monitoring the link, it would only check its data base after receiving a mobility notification, in turn notifying the applications whose ranges had changed.

7 Conclusions

Even though extending the Internet over wireless links is rather straightforward, application performance over wireless links using the traditional Internet protocols has been disappointing due to channel impairments and adverse interactions between protocol layers. Different applications favor different error control approaches, thus we developed a link layer architecture that provides multiple services simultaneously over a single link, allowing the most appropriate link service to be used by each flow. Our approach is easy to optimize for each underlying wireless link and efficient to operate. It can be locally deployed, transparently to the rest of the Internet, and it is easy to extend to address future requirements. It can be incorporated into the Internet QoS architecture in three stages. First, in the current best-effort only Internet, we can employ heuristic classifiers to select services provided over individual multi-service links to enhance the performance of well-known applications. Second, in the context of the Differentiated Services architecture, which focuses on end-to-end congestion control, multi-service link layers can provide application dependent error control, respecting higher layer scheduling decisions despite introducing recovery overhead. Finally, multi-service link layers can support an advanced QoS application interface, offering dynamic service discovery and synthesis.

References

1. Stevens, W.: TCP slow start, congestion avoidance, fast retransmit, and fast recovery algorithms. RFC 2001 (1997)
2. Nguyen, G.T., Katz, R.H., Noble, B., Satyanarayanan, M.: A trace-based approach for modeling wireless channel behavior. Proc. of the Winter Simulation Conference (1996)
3. Karn, P.: The Qualcomm CDMA digital cellular system. Proc. of the USENIX Mobile and Location-Independent Computing Symposium (1993) 35–39
4. Balakrishnan, H., Padmanabhan, V.N., Seshan, S., Katz, R.H.: A comparison of mechanisms for improving TCP performance over wireless links. Proc. of the ACM SIGCOMM (1996) 256–267
5. Badrinath, B.R., Bakre, A., Imielinski, T., Marantz, R.: Handling mobile clients: A case for indirect interaction. Proc. of the 4th Workshop on Workstation Operating Systems (1993) 91–97
6. Kent, S., Atkinson, R.: IP encapsulating security payload (ESP). RFC 2406 (1998)
7. DeSimone, A., Chuah, M.C., Yue, O.: Throughput performance of transport-layer protocols over wireless LANs. Proc. of the IEEE GLOBECOM (1993) 542–549
8. Xylomenos, G., Polyzos, G.C.: Internet protocol performance over networks with wireless links. IEEE Network **13** (1999) 55–63
9. UCB/LBNL/VINT Network Simulator – ns (version 2). Source code available at `http://www-mash.cs.berkeley.edu/ns`
10. Xylomenos, G.: Multi Service Link Layers: An Approach to Enhancing Internet Performance over Wireless Links. Ph.D. Dissertation, Dept. of Computer Science and Engineering, University of California, San Diego (1999)
11. Nanda, S., Goodman, D.J., Timor, U.: Performance of PRMA: a packet voice protocol for cellular systems. IEEE Transactions on Vehicular Technology **40** (1991) 584–598
12. Brady, P.T.: Evaluation of multireject, selective reject, and other protocol enhancements. IEEE Transactions on Communications **35** (1987) 659–666
13. Xylomenos, G., Polyzos, G.C.: Link Layer Support for Quality of Service on Wireless Internet Links. IEEE Personal Communications **6** (1999) 52–60
14. Blake, S., Black, D., Carlson, M., Davies, E., Wang, Z., Weiss, W.: An architecture for Differentiated Services. RFC 2475 (1998)
15. Golestani, S.: A self-clocked fair queueing scheme for broadband applications. Proc. of the IEEE INFOCOM (1994) 636–646
16. Zhang, L., Deering, S., Estrin, D., Shenker, S., Zappala, D.: RSVP: A new resource reservation protocol. IEEE Network **7** (1993) 8–18
17. Clark, D., Shenker, S., Zhang, L.: Supporting real-time applications in an integrated services packet network: architecture and mechanism. Proc. of the ACM SIGCOMM (1996) 243–254
18. Shenker, S., Breslau, L.: Two issues in reservation establishment. Proc. of the ACM SIGCOMM (1995) 14–26
19. Perkins, C.: IP mobility support. RFC 2002 (1996)

20. Ferrari, D., Verma, D.C.: A scheme for real-time channel establishment in wide-area networks. IEEE Journal on Selected Areas in Communications **8** (1990) 368–379
21. Johnson, D.B., Maltz, D.A.: Protocols for adaptive wireless and mobile networking. IEEE Personal Communications **3** (1996) 34–42
22. Noble, B, Price, M., Satyanarayanan, M.: A programming interface for application-aware adaptation in mobile computing. Proc. of the 2nd USENIX Symposium on Mobile and Location-Independent Computing (1995) 57–66

Enhancing the General Packet Radio Service with IP QoS Support

Giannis Priggouris, Stathes Hadjiefthymiades, Lazaros Merakos

Communication Networks Laboratory, Department of Informatics, University of Athens
Panepistimioupolis, Ilisia, Athens 15784, Greece
iprigg@voyager.di.uoa.gr, shadj@di.uoa.gr, merakos@di.uoa.gr

Abstract. The General Packet Radio Service (GPRS) is the current enhancement in the GSM infrastructure, capable of handling Internet Protocol (IP) traffic for mobile computing and communications. A major deficiency of the current GPRS specification is the lack of adequate IP Quality of Service (QoS) support. In this paper a QoS aware architecture for enhancing the GPRS core network with IP QoS support is presented. The proposed scheme is based on the Integrated Services (Intserv) architecture, processed in the IETF during the past years. The applicability of the proposed architecture as well as the required signalling enhancements and modifications in the components of the GPRS architecture are discussed. To identify the applicability of our proposal in a real GPRS network, a simulation model of the proposed IntServ architecture was developed in order to quantify the effect that signalling overhead has on the GPRS operation and performance. The obtained simulation results show that the proposed scheme demonstrated good scalability, even for large user populations.

1 Introduction

The General Packet Radio Service (GPRS), [1], [3] [5], [6], [8], is the current enhancement in the GSM infrastructure (GSM phase 2+) developed to satisfy the need for a true packet data service and offer Internet services to mobile users. GPRS offers packet switched data services to the mobile user, at speeds ranging from a few kbps to over 100 kbps, while circuit switched services, such as voice telephony, will still use the standard GSM infrastructure. Thus, GPRS does not aim to replace the current GSM circuit switched infrastructure but, rather, to co-exist with it. The choice of which service to be invoked at a certain circumstance, circuit or packet switched, depends on the type of the application.

An important issue in the Internet and, consequently, in every network connected to it, is the support for multimedia applications (video, voice). These applications have specific requirements in terms of delay and bandwidth, which challenge the original design goals of IP's best effort service model, and call for alternate service models and traffic management schemes that can offer the required QoS. To this end, two QoS architectures have emerged in the IETF: IntServ, which provides end-to-end QoS on a per-flow basis, and DiffServ, which supports QoS for traffic aggregates, [18].

This paper focuses on the first of the two aforementioned IP QoS schemes namely the Integrated Services. It aims at overcoming the main limitations in the current GPRS architecture that are briefly outlined below:

1. GPRS can differentiate QoS only on the basis of the IP address of a mobile station (MS) but not on the basis of IP flows, and
2. The GPRS core network uses IP tunnels, which makes the applicability of IP QoS schemes troublesome. Solutions are proposed to the problem of establishing QoS reservations across the GPRS core network, and the required signalling enhancements and modifications in the components of the GPRS architecture are identified.

In our proposal, the need for frequent refreshing of state information, and the use of RSVP [4] and additional signalling, which introduce traffic overhead, raise questions on the feasibility, scalability and performance. To address this issue, a system model, which includes models of the GPRS cellular infrastructure, network traffic and user movement, was developed.

The organisation of the rest of the paper is as follows. In Section 2, we provide an overview of the GPRS mechanisms for QoS provision to GSM subscribers, and identify their limitations in handling IP QoS traffic. In Section 3 our proposed IntServ architectural solution is presented and analysed. The simulation model for the architecture and the obtained results are presented and discussed in Section 4. Finally, Section 5 contains our conclusions.

2 GPRS QoS

The GPRS architecture comprises a set of GPRS Supporting Nodes (GSNs), i.e., nodes equipped with a GPRS compliant protocol stack. Two types of GSNs are defined: Supporting GSN (SGSN) and Gateway GSN (GGSN). The latter acts as a logical interface to external Packet Data Networks (PDNs) and, consequently, to the global Internet. The former is responsible for servicing the Mobile Stations (MS) currently located within its control area.

In GPRS, a specific QoS profile (part of the "PDP Context Profile" [6]) is being assigned to every subscriber upon his attachment to the network. This profile contains information such as:

- traffic precedence class,
- delay class,
- reliability class,
- peak throughput class, and
- mean throughput class.

Traffic precedence class may be of high, normal and low priority. Four delay classes and five reliability classes are defined. Nine peak throughput (i.e., 8, 16, 32, 64, 128, 256, 512, 1024, 2048 kbps) and 19 mean throughput classes (Best effort ... 111 kbps) are defined in the GPRS specifications.

The GPRS QoS profile can either be requested by the mobile user, during a phase known as PDP context activation, or, if no profile is requested, a default one, assigned to the user upon his subscription, is being activated. This default QoS profile is defined in the Home Location Register (HLR), along with other subscriber information. SGSN is the entity responsible for fulfilling the QoS profile request of the MS. After the activation of a certain profile no modification by the MS is allowed as long as the current one is active. If the MS wishes to modify its QoS profile, it should detach from the network, and then attach again specifying a new QoS profile. SGSN, on the other hand, can modify an active QoS profile at any time depending on the network load and the resources available.

Although a QoS profile is defined, this profile does not differentiate treatment of data flows within the core GPRS network (i.e., between the SGSN and the GGSN). All flows within the network are treated uniformly (best effort IP) and the aforementioned QoS parameters are used to allocate resources outside the core network. Thus, after the activation of a certain QoS profile, the MS is responsible for shaping its traffic to the negotiated QoS and the GGSN for restricting data flows to the MS based on its QoS profile. This implementation can be considered as a network with guaranteed bandwidth on the access interface, but with no provision for allocating resources along the GPRS core network. Clearly, such a scheme can work if ingress and egress nodes are directly linked. However, in a realistic topology, like the one shown in Fig. 1, things are quite different, and if no QoS mechanism is to be used, congestion would soon degrade the delivered performance.

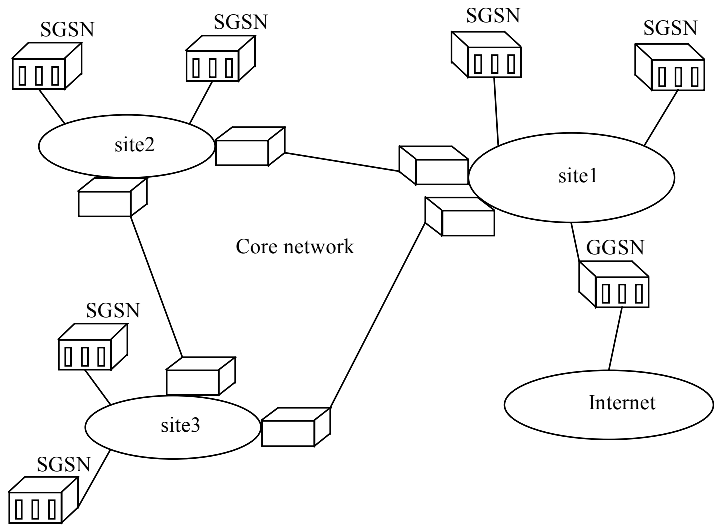


Fig. 1. GPRS backbone network

Communication between SGSN and GGSN is based on IP tunnels [13]. That means that standard IP packets, as soon as they reach a GSN (SGSN or GGSN), are encapsulated in new IP packets (i.e., a new IP header is added) and routed accordingly. The GPRS Tunnel Protocol (GTP) [6], implemented at both the GGSN and the SGSN, is used for this encapsulation. The first point, within the GPRS network, where IP packets are interpreted (i.e., header information is accessible) is the GGSN. This restriction implies a major difficulty in applying the IntServ IP QoS framework [4], within the GPRS network. An RSVP message, to say, would be transported transparently within the GPRS network towards the GGSN or vice versa causing no reservations at intermediate nodes (routers). Another problem arising from this tunnel - based communication is that different data flows addressed to the same terminal (same IP address) are treated in the same manner. Routing, within the GPRS core, is based on the IP addresses of GSNs. Discrimination of packets in GSNs is based on tunnel identifiers (TIDs). There is a one-to-one mapping between MS IP addresses and TIDs (see Fig. 2). Such implementation is not well aligned with the IntServ framework, where micro-flows are a basic concept. In the following section, we present and analyse an architectural solution to overcome the limitations imposed by the current GPRS architecture.

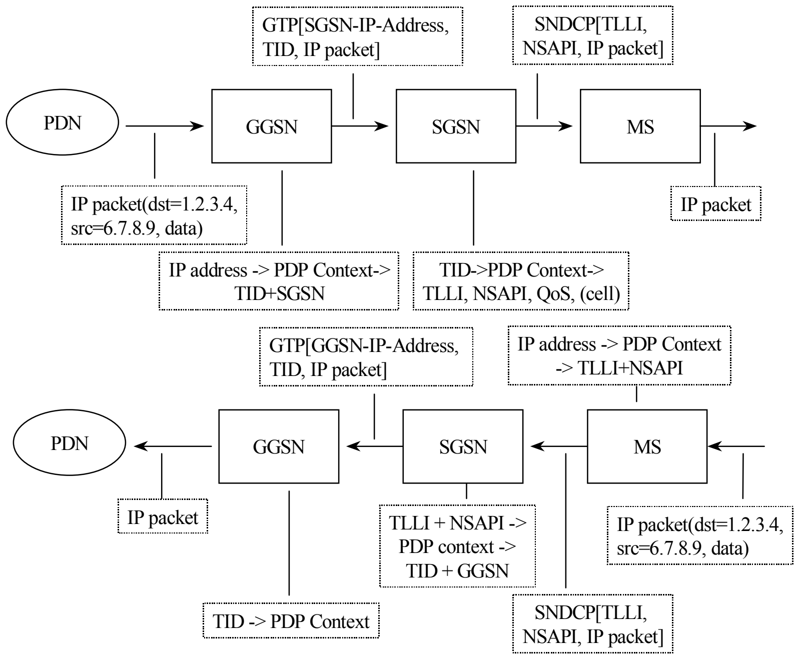

GTP[SGSN-IP-Address, TID, IP packet]
SNDCP[TLLI, NSAPI, IP packet]
PDN
GGSN
SGSN
MS
IP packet(dst=1.2.3.4, src=6.7.8.9, data)
IP packet
IP address -> PDP Context-> TID+SGSN
TID->PDP Context-> TLLI, NSAPI, QoS, (cell)
GTP[GGSN-IP-Address, TID, IP packet]
IP address -> PDP Context -> TLLI+NSAPI
PDN
GGSN
SGSN
MS
IP packet
TLLI + NSAPI -> PDP context -> TID + GGSN
IP packet(dst=1.2.3.4, src=6.7.8.9, data)
TID -> PDP Context
SNDCP[TLLI, NSAPI, IP packet]

Fig. 2. Data flows in GPRS

3 An Architecture in GPRS

In this section, our IntServ – based architecture is presented. It should be stressed, from the beginning, that our effort to enhance the present GPRS QoS mechanism is focused on the GGSN, as this is the first node within the network, where IP packets are interpreted.

Two main issues arise in our effort to enhance the current GPRS architecture with IP QoS awareness. Firstly, the establishment of QoS reservations across the GPRS core and, secondly, the mapping of RSVP signalled[1] QoS requirements to the GPRS QoS characteristics applicable in the MS - SGSN interface (i.e., the Context Profile).

RSVP tunnelling [15], [16], could be used to establish QoS reservation across the GPRS core. Original (i.e., end-to-end) RSVP signalling messages pass transparently (i.e., tunnelled) through the GPRS network, but additional RSVP signalling is used to create and maintain reservations at each node (router) of the backbone network. The Renty and Rexit of the RSVP tunnel are the GGSN and SGSN respectively (for traffic destined to the MS). The roles are reversed for traffic destined to the fixed network/GGSN. All traffic pertaining to a particular MS will be aggregated in a single RSVP tunnel (both in the uplink and downlink directions). Since RSVP tunnelling specifies the use of source UDP port numbers for distinguishing different reservations across the tunnelled part, there should be a one-to-one mapping between GTP TIDs and UDP port numbers. GGSN is the co-ordinating entity for all reservations meaning that it triggers reservations, for both the uplink and the downlink. Its peer entity, SGSN, is not capable of directly reacting to RSVP signals since it only handles encapsulated packets. Therefore, GGSN, being aware of the transmission of a RESV/PATH signal by the MS, signals SGSN appropriately (through Trigger_RESV and Trigger_PATH, shown in Fig. 3 and Fig. 4) so as to proceed with the required reservations (i.e., complete a reservation through a RESV or initiate the relevant procedure through a PATH). The Message Sequence Charts (MSC) [2] for reservation establishments, across the GRPS core, in response to incoming/outgoing PATH signals, is shown in Fig. 3 and Fig. 4 respectively.

The characteristics (i.e., Tspec) of the RSVP tunnel are dictated by the characteristics of the individual sessions that are nested in the tunnel (e.g., a simple addition of peak rates, or use some kind of equivalent bandwidth metric).

As shown in Fig. 3 and Fig. 4, following the successful establishment of new reservations across the GPRS core, the GPRS QoS profile (of the particular connection) needs to be modified to reflect the new situation. This task should be undertaken by the GSN that accepted the requested RSVP reservation. In the case of an outgoing PATH (see Fig. 4), such operation is realised using the standard signalling of the PDP Context Modification Procedure defined in the GPRS specification [6]. The signals used are UpdatePDPContextRequest and UpdatePDPContextResponse. In the incoming PATH case, the QoS change should be initiated by the GGSN. Two new signals are introduced, namely the ChangePDPContextRequest and ChangePDPContextResponse to fulfil the new requirements. In both cases, a notification is also transmitted to the MS using the ModifyPDPContextRequest and ModifyPDPContextAccept signals. Context Profile modifications are performed only if the RSVP signals are referring to new reservations/sessions and not refreshes of existing ones.

[1] Either by the MS or by a fixed node outside the Public Land Mobile Network (PLMN).

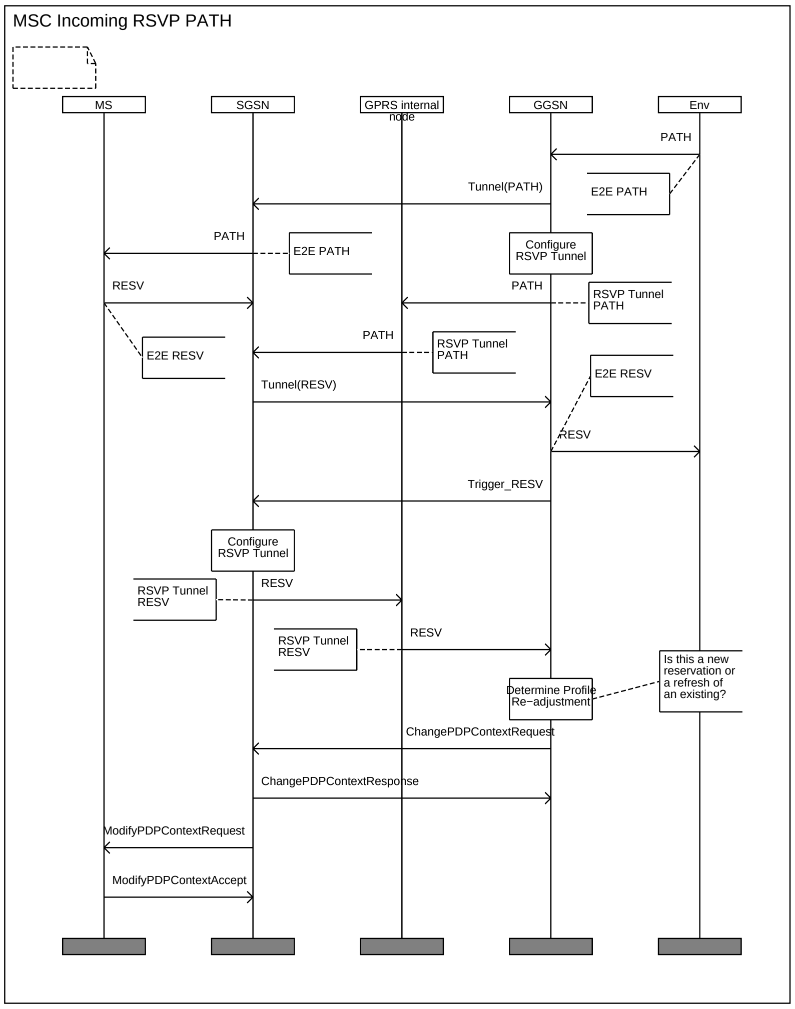


Fig. 3. MSC for incoming PATH (successful reservation)

A very important issue in the discussed scheme is the mapping between RSVP information elements, pertaining to the tunnel session, and GPRS QoS classes. Many mapping approaches can be thought of, but this is mostly an operator's decision. In [11], a mapping of the Best Effort and Controlled Load IP services to GPRS QoS Classes has been proposed.

Finally, we note that for the implementation of the IntServ solution described above, only the GGSN should have to be changed significantly (enhancement of existing protocols), while the rest of the nodes need, simply, to be upgraded with RSVP functionality.

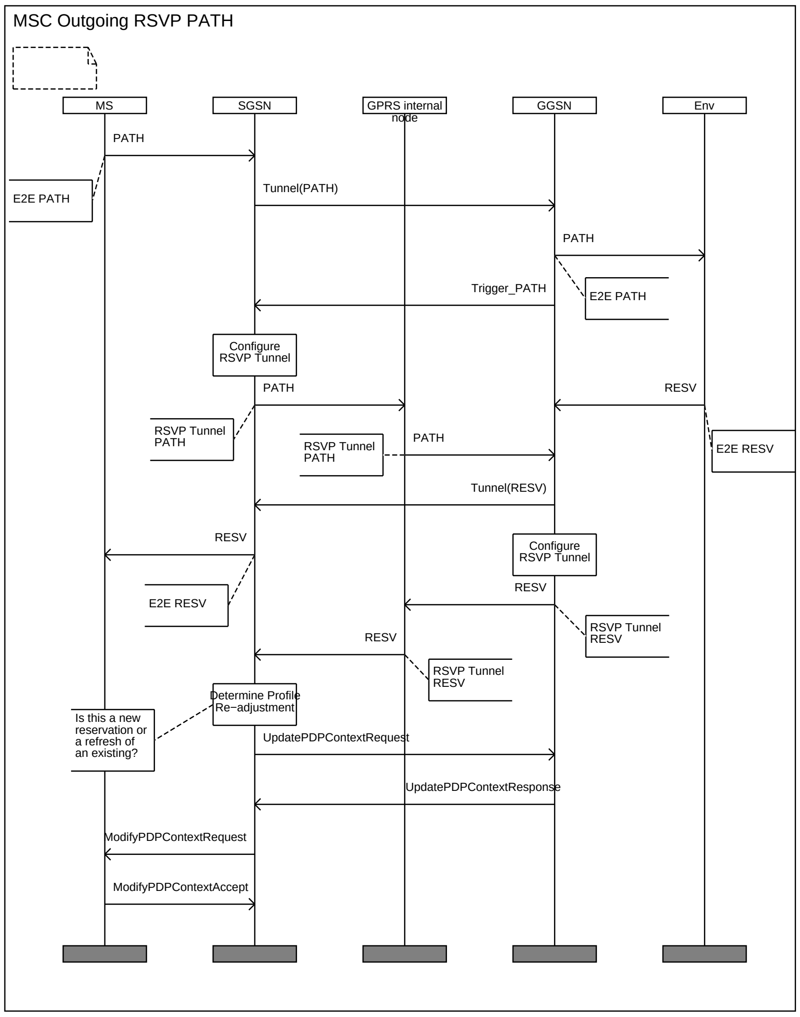


Fig. 4. MSC for outgoing PATH (successful reservation)

4 Simulation of the Proposed Architecture

In the scenario described in the previous section, the need for frequent refreshing of state information, and the use of RSVP [4] and additional signalling, which represent traffic overhead, raise questions on the feasibility, scalability and performance of the proposed solution. To address this issue, a simulation model of the proposed architecture was developed using the PARSEC discrete event simulation package [10]. This model includes modelling of the GPRS cellular infrastructure, network traffic and user movement. The most important metrics that we monitored through the simulation environment were:

- the number of handovers observed while applications were active,
- the number of failed reservations, i.e., reservations that were dropped due to handovers towards loaded SGSNs, or reservation requests that were rejected in loaded SGSNs, from the very beginning,
- the signalling overhead caused in the GPRS core as a result of the implemented reservation scheme.

Below, we provide a brief description of the developed simulation models.

4.1 Modeling of the Cellular Infrastructure

The considered network model comprises 5 SGSNs and a single GGSN for the interconnection of MS to the Internet. Each SGSN is directly connected to the GGSN with 2 Mbps Frame Relay links [6]. Each SGSN is handling a number of MSs distributed over a number of cells/base stations. The modelling of the cellular environment focused on the GPRS backbone network, since it is the only part of the architecture affected by the introduction of RSVP. The GSM radio subsystem was not taken into account. IP traffic was introduced in the model through 12 servers located outside the GPRS core. Such servers were 4 FTP servers, 4 video servers and 4 voice servers. The simulated architecture is shown in Fig. 5.

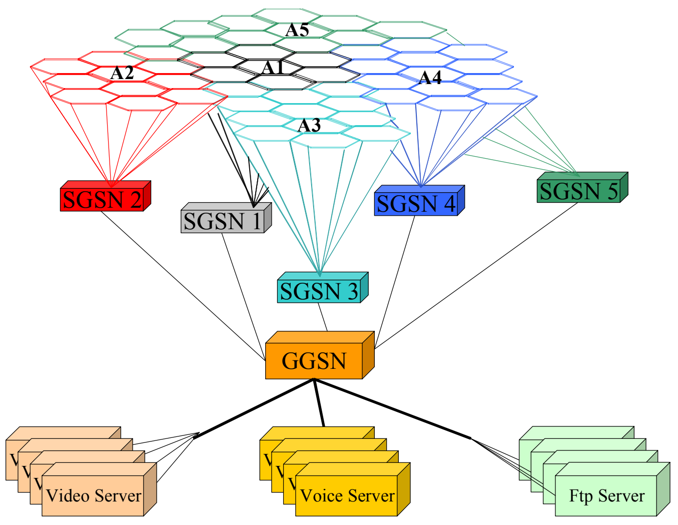


Fig. 5. Simulated architecture

The placement of the application servers in the architecture of Fig. 5 does not necessarily imply that they can be found one hop away from the GGSN within the IP backbone. In the general case, multiple network nodes may constitute the path from the GGSN to some application server. To achieve this differentiation, the times taken for the completion of requests are modelled as random variables in accordance with the published statistics for Internet traffic.

4.2 Network Traffic Modelling

The total simulation time was 4 hours. During the first 30 minutes, the total number of MSs, monitored through the simulation, is becoming active (i.e., perform the GPRS attach procedure). Initially the MS population is evenly distributed among the 5 SGSNs. In the simulation environment, each active MS is frequently performing handover to areas controlled by other SGSNs. The MS movements monitored by the simulation program are not simple changes of base stations but rather Routing Area (RA) updates resulting in inter-SGSN handovers. The relocation of MS in adjacent SGSN is performed stochastically. Relocations towards A1 are more likely to happen than relocations to the surrounding control areas. Hence, A1 handles a considerable percentage of user population simulating a real life environment (e.g., city centre). The adopted transition probabilities for the various control areas are shown in Fig. 6.

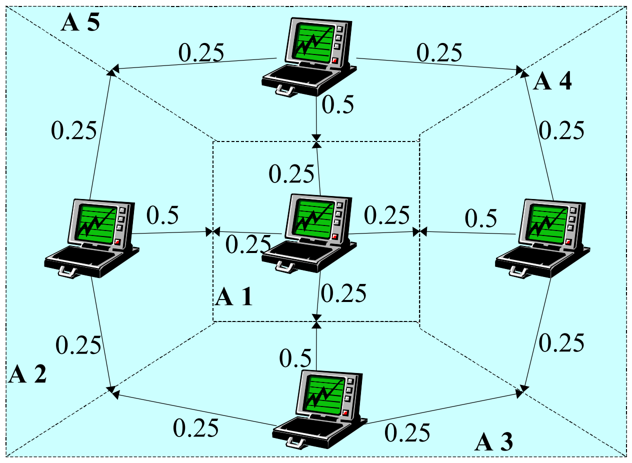


Fig. 6. Probabilities for inter-SGSN handovers

The Intra-SGSN residence time (the time spent by the user within the control area of the current SGSN) is modelled by a random variable R, whose probability distribution $F_R(\cdot)$ is as shown below.

$$F_R(r) = f \cdot G_f(r) + (1-f)G_s(r) \tag{1}$$

In (1), f denotes the fraction of the fast moving users over the entire user population ($0\leq f\leq 1$); $G_f(\cdot)$ and $G_s(\cdot)$ denote the probability distributions of a fast and a slow moving user, respectively. Both $G_f(\cdot)$ and $G_s(\cdot)$ are assumed truncated Normal distributions [17] to avoid negative and unrealistically small residence times and are as given below.

$$G_f(r) = \begin{cases} \dfrac{N_f(r) - N_f(t_f)}{1 - N_f(t_f)}, \ t_f < r < +\infty \\ \\ 0, \ -\infty < r \le t_f \end{cases} \tag{2}$$

$$G_s(r) = \begin{cases} \dfrac{N_s(r) - N_s(t_s)}{1 - N_s(t_s)}, \ t_s < r < +\infty \\ \\ 0, \ -\infty < r \le t_s \end{cases} \tag{3}$$

where $N_f(\cdot)$, $N_s(\cdot)$ are normal distributions, and t_f, t_s are non-negative threshold parameters.

While roaming, each active MS spawns QoS applications, which cause RSVP requests to propagate within the GPRS core. QoS applications refer to video, voice and FTP applications, each one with specific requirements in terms of the requested resources and duration [9], [12]. In each MS applications are invoked stochastically with probabilities as shown in Table 1. In the same table the requested bandwidth for each application is, also, displayed. Note that, in order to make our model more realistic and prevent the individual user from monopolising bandwidth each MS can simultaneously activate up to three (3) QoS applications.

Table 1. Probabilities and resource requirements for different application types

Application Type	Session Activation Probability (%)	Requested resources (Kbps)
Best Effort (telnet, web)	0.5	0
Video	0.1	56
Voice	0.15	28.8 (Streaming e.g. RealAudio)
		16 (VoIP)
FTP	0.25	10

Each MS is initially granted a default QoS Profile that allows the exchange of Best Effort traffic. In the course of simulation, this Profile changes dynamically depending on the needs of user applications. Each time a new application is activated, a resource reservation request is passed on to the network. Such request is signalled through RSVP (i.e., the MS produces only GPRS and RSVP signalling). Provision has been taken to simulate "rush hour" conditions by varying the frequency of QoS applications instantiations/requests throughout the simulation period.

4.3 Simulation Results

We have performed simulations for three, different MS populations: 300, 400, and 500 MSs. The principal criteria for the assessment of the results of the simulation process are:

- a low percentage of failed reservations, and
- a relatively low signalling overhead introduced by the extra RSVP signalling used for reservations.

Fig. 7 and Fig. 8 present the simulation results for the population of 400 MSs. However results are quite alike in the case of GPRS networks with 300 and 500 users. Fig. 7 shows the changes in the number of attached MSs per SGSN throughout the simulation period for the 400 MS population scenario. Lastly, in Fig. 8, we plot the mean and maximum amount of reserved resources monitored in each MS individually. Expected values are also displayed in the same figure. It is straightforward from this last figure that the maximum reservation for each MS varies at values around 60 kbps while cases of extensive reservations (i.e., over 100 kbps) do exist, but are rather rare. The corresponding mean reservation for each MS is considerably lower (about 24 kbps). Note that in Fig. 8 the term “expected” denotes the averaging of the respective metrics over the entire population of MSs. “Active” means that only those time periods when QoS connections existed were taken into consideration for the calculation of the respective value.

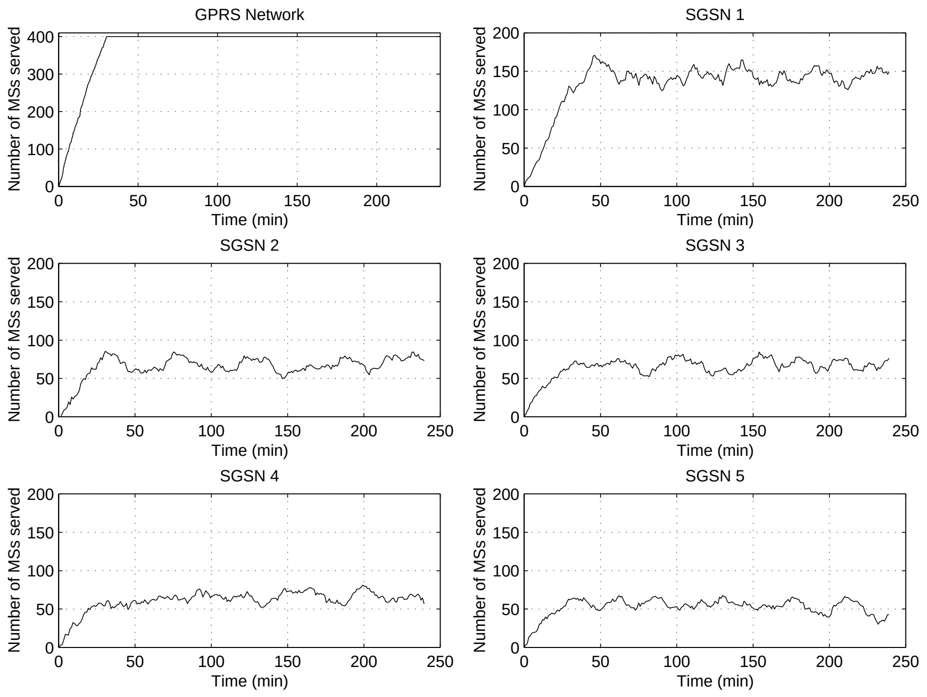


Fig. 7. Simulation results for the population of 400 MSs (1)

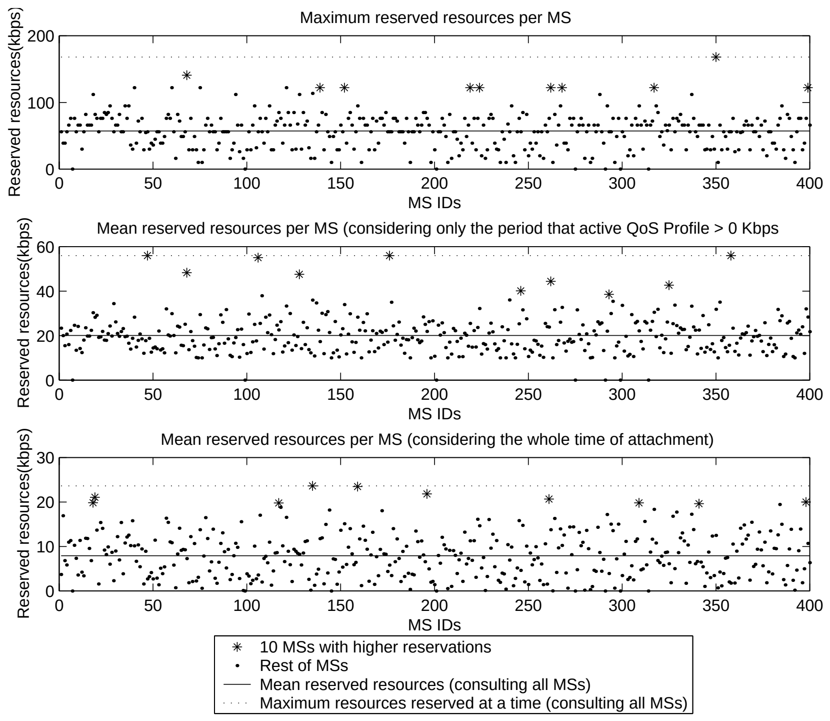


Fig. 8. Simulation results for the population of 400 MSs (2)

Table 2. Percentage of handovers with active connections

Population	Handovers		
	Total	Active	Active (%)
300	4275	1713	40.07
400	5647	2113	37.41
500	7087	2673	37.72

In Table 2, we present the number and the percentage of handovers that involved active connections/reservations over the total number of registered handovers, for all, three simulation scenarios. Handovers with active QoS connections (RSVP/IntServ) cause significant problems as discussed in [7]. In the same Internet Draft, various proposals to overcome this problem are also presented.

Table 3. Failed reservations

Population	Requested reservations	Failed reservations[2]	Failed reservations (%)	Mean time between failed reservations (secs)
300	3848	11	0.3	205
400	4921	89	1.8	41.6
500	6016	216	3.6	19.8

Table 3 lists the logged failed reservations for the three population scenarios. It should be noted that the vast majority of these failed reservations were observed in a time period of one hour, which, in turn, is well approximated by the simulated "rush hour".

The last important metric that we monitored through the simulation environment was the overhead introduced by the RSVP tunnelling technique, and the TriggerResv/Path signalling messages. Additionally, we evaluated the sensitivity of the signalling overhead to changes in the RSVP tunnel refresh period[3] (R). Table 4 summarises our findings.

Table 4. Signaling overheads

Population	R (secs)	Signaling overhead (%)	Traffic overhead (IPv4) (%)	Bandwidth Overhead (IPv4) (kbps)	Traffic overhead (IPv6) (%)	Bandwidth overhead (IPv6) (kbps)
300	30	84.14	2.11	16.55	2.41	23.70
	60	52.38	1.95	12.08	2.18	17.01
400	30	85.01	2.11	20.33	2.41	29.09
	60	52.58	1.96	16.25	2.19	23.03
500	30	84.54	2.13	24.73	2.45	35.50
	60	51.81	1.97	19.61	2.21	27.89

Signaling overhead denotes the additional RSVP signalling, in terms of messages, used to establish RSVP tunnels. Traffic overhead is calculated by dividing RSVP signalling volume by the total IP traffic exchanged. Analogous calculation is performed for bandwidth. Traffic and bandwidth overheads were calculated for both an IPv4 and an IPv6 GPRS backbone. The results for both versions were quite similar, with the IPv4 architecture demonstrating a slightly better behaviour. Another issue that should be noted is that although reduction of the tunnel refresh period (R) affects greatly the signalling overhead (over 30% reduction), it has no significant impact on the traffic overhead (less than 0.25% reduction). However, traffic

[2] Failed reservations = dropped reservations + rejected requests

[3] Although the tunnel RSVP session is triggered from the End-to-End RSVP session, it constitutes an independent RSVP session. Thus, its refresh period can be different from the corresponding End-to-End.

overheads in all cases are quite acceptable, as they do not surpass the 2.5% of the total IP traffic exchanged.

5 Conclusions

We have presented a new architectural scheme on how the GPRS architecture could be enhanced with IP QoS awareness.

GPRS is an overlay network to the GSM infrastructure. The importance of GPRS for the emerging area of mobile computing is extremely high. In this respect, GPRS is being considered as the core network technology for UMTS Phase 1.

In GPRS, QoS guarantees to IP traffic are provided only per MS IP address hence, any kind of differentiation of IP flows is not possible. In the core network, the applicability of IP QoS schemes is also troublesome due to the use of IP tunnels. To the best of our knowledge, no previous work has been reported in the same area apart from the suggestions given in [14] and [11] that are quite aligned with what is presented in this paper.

The proposed solution tries to alleviate some of the drawbacks discussed above. The per IP address QoS scheme forces the adoption of coarse-grained QoS solutions (e.g., Tunnels) in the GPRS core. The suggested solution seems to suffer from scalability, since, it capitalises on a soft-state model but also introduces new signals. The soft–state model requires constant refreshing resulting to signalling overhead which has a negative impact on the network load.

To quantify the effect that signalling overhead has on the IntServ system operation and performance, we have developed a simulation model of the proposed scenario. The obtained simulation results have shown that the proposed IntServ architecture demonstrated good scalability, even for large user populations, mainly because of the use of the RSVP tunnelling scheme for the establishment of reservations within the GPRS core. RSVP tunnels treat the flows towards a particular MS as aggregated traffic, causing a significant reduction in the number of RSVP messages. RSVP signalling, as a fraction of the total IP traffic, is small (<2.5%), and can be reduced even more with the appropriate adjustment of the RSVP tunnel refresh period. However, even without this adjustment the total traffic overhead introduced in the GPRS core was found more than acceptable.

6 References

1. Bettstetter, C., Vögel, H-J., Eberspächer, J.: GSM Phase 2+ General Packet Radio Service GPRS: Architecture, Protocols, and Air Interface. IEEE Communications Surveys, Vol.2, No.3, (1999)
2. Bræk, R., Haugen, O.: Engineering Real Time Systems. Prentice Hall (1993)
3. Brasche, G., Walke, B.: Concepts, Services, and Protocols of the New GSM Phase 2+ General Packet Radio Service, IEEE Communications Magazine (1997)
4. Braden, R., Zhang, L., Berson, S., Herzog, S., Jamin, S.: Resource ReSerVation Protocol (RSVP) - Version 1 Functional Specification, IETF RFC 2205, Network Working Group, (1997)

5. Cai, J., Goodman, D.J.: General Packet Radio Service in GSM, IEEE Communications Magazine, (1997)
6. Draft EN 301 344 V6.1.1, Digital cellular telecommunications system (Phase 2+); General Packet Radio Service (GPRS); Service Description; Stage 2 (GSM 03.60 version 6.1.1 Release 1997), ETSI (1998)
7. Fankhauser, G., Hadjiefthymiades, S., Nikaein, N., Stacey, L.: RSVP Support for Mobile IP Version 6 in Wireless Environments, Internet Draft, <draft-fhns-rsvp-support-in-mipv6-00>, Mobile IP Working Group, (1998).
8. Granbohm, H., Wiklund, J.: GPRS - General packet radio service, Ericsson Review No.2, (1999)
9. Lee, B.O., Frost, V.S., Jonkman, R.: NetSpec 3.0 Source Models for telnet, ftp, voice, video and WWW traffic, Technical Report ITTC-TR-10980-19, Information and Telecommunication Technology Center, The University of Kansas, (1997)
10. Meyer, R. A.: PARSEC User Manual, Release 1.1, UCLA Parallel Computing Laboratory, (1999)
11. Mikkonen, J., Turunen, M.: An Integrated QoS Architecture for GSM Networks, Proceedings of ICUPC '98, Florence, Italy, (1998)
12. Paxson, V., Floyd, S.: Wide-Area Traffic: The Failure of Poisson Modelling, IEEE/ACM Transactions on Networking, Vol.3, No. 3, (1995)
13. Perkins, C.: IP Encapsulation within IP, RFC 2003, (1996)
14. Puuskari, M.: Quality of Service Framework in GPRS and Evolution Towards UMTS, proceedings of the 3rd European Personal Mobile Communications Conference, Paris, (1999)
15. Terzis, A., Krawczyk, J., Wroclawski, J., Zhang, L.: RSVP Operation Over IP Tunnels, Internet-Draft, <draft-ietf-rsvp-tunnel-04.txt>, (1999)
16. Terzis, A., Srivastava, M., Zhang, L.: A Simple QoS Signalling Protocol for Mobile Hosts in the Integrated Services Internet, Proceedings of IEEE Infocom '99, New York, USA (1999)
17. Woodward, M. E.: Communication and Computer Networks: Modelling with Discrete-Time Queues, IEEE Computer Society Press, (1994)
18. IEEE Network Magazine, Special Issue on Integrated and Differentiated Services for the Internet, Vol. 13, No. 5, (1999)

Genetic Algorithm for Mobiles Equilibrium Applied to Video Traffic

Mohamed Moustafa[1], Ibrahim Habib[1], and Mahmoud Naghshineh[2]

[1] The City College of the City University of New York, Electrical Engineering Department, Convent Av. & 138th. St, New York, NY 10031
{Moustafa, eeiwh}@ee-mail.engr.ccny.cuny.edu
[2] IBM T.J. Watson Research Center,
30 Saw Mill River Rd, Hawthorne, NY 10532
mahmoud@us.ibm.com

Abstract. In a CDMA network, resource allocation is critical in order to provide suitable signal quality for each user and achieve channel efficiency. The third-generation mobile communication systems (ITU/IMT-2000) must be designed to support wideband services at bit rates as high as 2 Mbps, with the same quality as fixed networks. Mobiles transmitted power has to be controlled to provide each user a reasonable connection while limiting the interference seen by other users. Transmitted rate has also to be controlled to avoid congestion. An adaptive protocol is proposed for controlling mobile calls transmitter power and rate cooperatively when previous work has focused on handling them separately. The active component of this scheme is called Genetic Algorithm for Mobiles Equilibrium (GAME). Based on an evolutionary computational model, the base station tries to achieve an adequate equilibrium between its users. Thereof, each mobile can send its traffic with a suitable power to support it over the different path losses and interference. In the mean time, its battery life is being preserved while limiting the interference seen by neighbors. A significant enhancement in signal quality and power level has been noticed through several experiments.

1 Introduction

During the past years, there has been an increasing demand for high-speed multimedia communication. This rapid contagious increase is being spread inside the wireless mobile community. Code Division Multiple Access (CDMA) has received a great amount of attention as a multiple access method for future mobile networks [9] since it allows a given transmission link to carry simultaneous virtual connections. The third-generation mobile communication systems, called International Mobile Telecommunications-2000 (IMT-2000) in the International Telecommunication Union (ITU) [13] must be designed to support wideband services at bit rates as high as 2 Mbps, with the same quality as fixed networks. We have seen recently several proposals [1] [15] that support third generation IMT-2000 systems by offering multiple voice/data connections with differing QoS requirements. Future generations will include high quality video as well.

Mobiles transmitted power has to be controlled to provide each user a reasonable connection while limiting the interference seen by other users [8]. Transmitted rate has also to be controlled to avoid congestion, which means dropping of quality below required levels. Previous work has focused on finding adequate power levels that maximize the received bit energy-to-noise density ratio (E_b/N_o) where the transmission rate of each user is fixed [6] [7]. Stochastic power control algorithms based on the mobile matched filter output is introduced in [12]. [14] proposed a circuit switched multirate DS-CDMA system based on a closed-form power control function. Recently, [3] presented an algorithm for controlling mobiles transmitter power levels following their time varying transmission rates.

This work presents an algorithm for controlling mobiles transmitter power and bit rate in tandem. The base station measures each mobile received signal quality, bit rate and power. Accordingly, it computes the proposed optimum power and rate vector and sends it back to all mobiles.

The remainder of the paper is organized as follows. In section 2, the optimization problem is formulated. The genetic algorithm (GA) engine is described in section 3. Some numerical results are presented in section 4 illustrating the deployment of the proposed GA. In section 5, we conclude.

2 Optimization Problem

In a CDMA network, resource allocation is critical in order to provide suitable Quality of Service (QoS) for each user and achieve channel efficiency. Many QoS measures, including bit error rate, depend on the E_b/N_o given by

$$\left(\frac{E_b}{N_o}\right)_i = \frac{G_{bi}p_i / R_i}{\left(\sum_{j\neq i}^{M} G_{bj}p_j + \eta\right) \Big/ W} \tag{1}$$

where W is the total spread spectrum bandwidth occupied by the CDMA signals. G_{bi} denotes the link gain on the path between mobile i and its base b. η denotes background noise due to thermal noise contained in W and M is the number of mobile users. The transmitted power of mobile i is p_i which is usually limited by a maximum power level as

$$0 \le p_i \le p_i^{\max}, \qquad for \quad 1 \le i \le M \tag{2}$$

R_i is the information bit rate transmitted by mobile i. This rate is bounded by the value, R_i^C , designated in the traffic contract once this user has been admitted into the system.

$$0 \le R_i \le R_i^C, \qquad for \quad 1 \le i \le M \tag{3}$$

An increase in the transmission power of a user increases its E_b/N_o, but increases the interference to other users, causing a decrease in their E_b/N_os. On the other hand, an increase in the transmission rate of a user deteriorates its own E_b/N_o. Controlling powers and rates of the users therefore amounts to directly controlling the QoS that is usually specified as a pre-specified target E_b/N_o value (θ). It can also be specified in terms of the outage probability, defined as the probability (Prb^{θ}) that the E_b/N_o falls below θ.

Thus, the objective here is to find a nonnegative power $\underline{P}=[p_1, p_2, ..., p_M]$ and rate $\underline{R}=[R_1, R_2, ..., R_M]$ vectors which satisfy the maximum power and rate constraints (2, 3) and maximize the function F proposed as

$$F = \frac{1}{M}\sum_{i=1}^{M} F_i^E \left(\omega_P F_i^P + \omega_R F_i^R\right) + \omega_H F^H \ , \qquad \textbf{(4)}$$

where F_i^E is a threshold function defined for user i as

$$F_i^E = \begin{cases} 1 & \left(E_b/N_o\right)_i \geq \theta_i \\ 0 & otherwise \end{cases} . \qquad \textbf{(5)}$$

While limiting the P-R search space to solutions that maximize the average QoS, through F_i^E , minimizing P is essential. Since low P means little interference and long battery life, F_i^P (6) gives credit to solutions that use minimum power and punishes others using high levels.

$$F_i^P = 1 - \frac{p_i}{p_i^{\max}} \ . \qquad \textbf{(6)}$$

Fulfilling a user requested bit rate is another objective as long as it is below its contract rate R_i^C . However, the future asked bit rate is unknown. Thus, the evaluation approximates the next asked bit rate as

$$R_i^a \cong \overline{R}_i^T + \overline{R}_i^D \ , \qquad \textbf{(7)}$$

where $\overline{R}_i^T$ is the average granted bit rate, and $\overline{R}_i^D$ is the average deficit bit rate (i.e., difference between asked and granted). The average is calculated over the previous control period T. F_i^R , as proposed in (8), grants priority to R_i in the vicinity of R_i^a

$$F_i^R = \begin{cases} R_i / R_i^a & R_i \leq R_i^a \\ \left(R_i^C - R_i\right) / \left(R_i^C - R_i^a\right) & R_i^a < R_i \leq R_i^C \\ 0 & R_i > R_i^C \end{cases} . \qquad \textbf{(8)}$$

To overcome what is known as the Near-Far effect [11], we have to ensure that the received E_b/N_os from all mobiles at the base station are in a narrow range. F^H , proposed in (9), evaluates the homogeneity of the received signal qualities by penalizing the solution whose received E_b/N_o components divergence is far away from its mean value.

$$F^H = \begin{cases} 1-5\left(\sigma_E/\overline{E}\right) & \sigma_E \le 0.2\overline{E} \\ 0 & otherwise \end{cases} \tag{9}$$

$\overline{E}$ is the average received E_b/N_o, and σ_E is the corresponding standard deviation.

ω_P, ω_R, and ω_H introduced in (4) are the nonnegative priority weights of each corresponding objective respectively. The priority weight indicates the relative importance of an objective over the other. For instance, if minimizing the transmitting mobile power is our highest priority, than ω_P would be the highest value. Therefore, the larger the F_i^P the more dominant the value of $\omega_P F_i^P$ in the overall value of F.

Therefore, the jointly optimal power and rate is obtained by solving the following optimization problem:

$$\max_{(P,R)\in\Omega} F(P,R) \tag{10}$$

where F is defined in (4) and the feasible set Ω is subject to power and rate constraints (2, 3).

3 GAME Engine

Genetic algorithms (GAs), as described in [2], are search algorithms based on mechanics of natural selection and natural genetics. In every generation, a new set of artificial creatures (chromosomes) is created using bits and pieces of the fittest of the old. GAs use random choice as a tool to guide a search toward regions of the search space with likely improvement. They efficiently exploit historical information to speculate on new search points with expected improved performance.

The **G**enetic **A**lgorithm for **M**obile **E**quilibrium core is a steady state GA [2] with the ability to stop its evolution after a timeout period being expired. As illustrated in Fig. 2, the inputs are collected from the users as their current rate $\underline{R}(t)$ and power $\underline{P}(t)$ vectors. Additional information is needed as well in the input like the required signal quality $\underline{\theta}$, the maximum possible power $\underline{P}^{max}$, and the contract rate granted on admission $\underline{R}^C$. The GAME starts by encoding $\underline{R}(t)$ and $\underline{P}(t)$ into chromosomes to form the initial population.

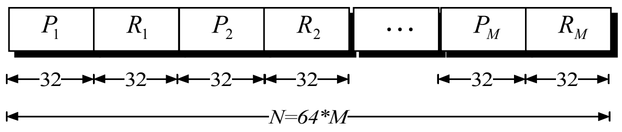


Fig. 1. GAME Chromosome Format: *N bits*, where *M* is the total number of mobiles controlled by the base station. Each Mobile occupies 64 bits to represent its power *P* and Rate *R*

The chromosome is a binary string of *N* digits. It encodes the (power, rate) values of all *M* mobiles as depicted in Fig. 1. Each mobile occupies 32 bits for its power and 32 bits for its rate.

F, defined in (4), is the fitness function used to evaluate chromosomes. The evolving cycle of reproduction-evaluation works until the stopping criterion is met. The base station forwards the new $\underline{R}(t+1)$ and $\underline{P}(t+1)$ vectors to the users.

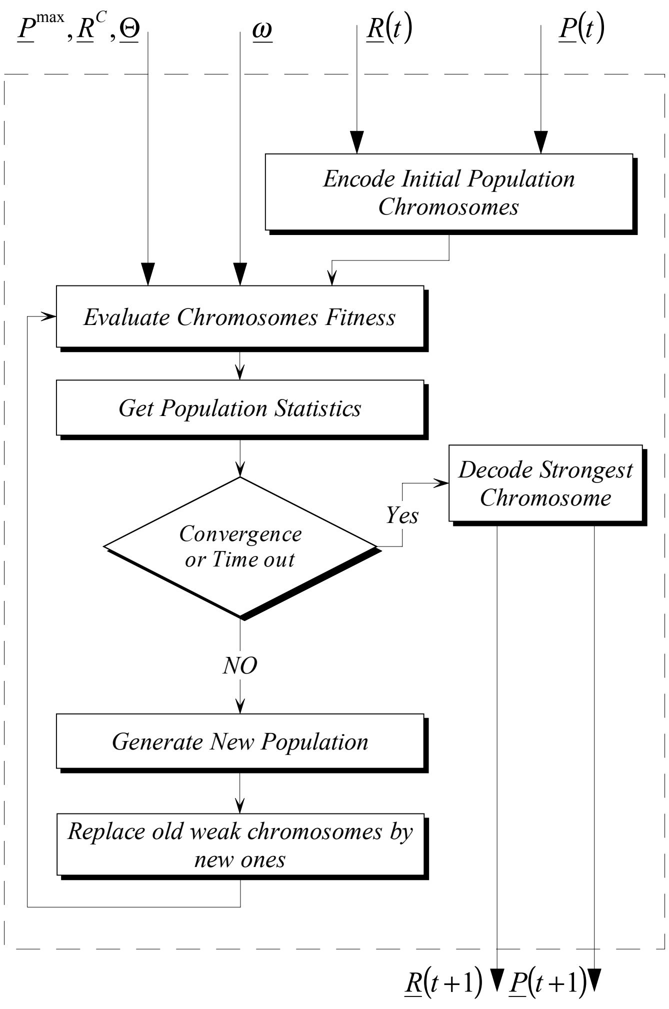

$\underline{P}^{\max}, \underline{R}^{C}, \Theta$
$\underline{\omega}$
$\underline{R}(t)$
$\underline{P}(t)$
Encode Initial Population Chromosomes
Evaluate Chromosomes Fitness
Get Population Statistics
Convergence or Time out
Yes
Decode Strongest Chromosome
NO
Generate New Population
Replace old weak chromosomes by new ones
$\underline{R}(t+1)$
$\underline{P}(t+1)$

Fig. 2. GAME Block Diagram

According to the fitness function, used to compare the solutions chromosomes, the fittest vectors $\underline{R}(t+1)$ and $\underline{P}(t+1)$ should be within the boundaries (2, and 3). In the mean time the granted bit rate should be near the requested one while the power level is minimum as possible. This solution also should be able to make each user surmounts its required E_b/N_o value (θ).

4 Numerical Results

Simulation for showing GAME behavior is done for a single cell, where the base is situated at its center. The cell radius is 2 km. Mobile users are distributed uniformly over the cell space. We adopt the ITU-R distance loss model [5]. The link gain, G_{ij}, is modeled as a product of two variables

$$G_{ij} = A_{ij} \times D_{ij} \tag{11}$$

A_{ij} is the variation in the received signal due to shadow fading, and assumed to be independent and log normally distributed with a mean of 0 dB and a standard deviation of 8 dB. The variable D_{ij} is the large-scale propagation loss, which depends on the transmitter and the receiver location, and on the type of geographical environments. Let d_{ij} be the distance in km between transmitter j and receiver i, the ITU-R formula yields the following path loss equation for a typical 3G CDMA system parameters [1].

$$10 \log D_{ij} = -76.82 - 43.75 \log d_{ij} \tag{12}$$

The center frequency is 2 GHz, antenna heights of the mobile and the base station are 1.5 and 30 m respectively. We assume that 20% of base station coverage area is occupied by buildings. The system bandwidth W can increase up to 20MHz and the background thermal noise density is –174 dbm/Hz.

Mobiles users with MPEG encoded video traffic [10] are used in the experiments. Encoder input is 384x288 pels with 12 bit color information. A 12 frames GOP pattern (IBBPBBPBBPBB) is generated at 25 frames/sec. Mean bit rate is 330 Kbps while the peak rate is 1.01 Mbps. Each mobile starts with 500 mW as initial transmitting power while its maximum power level is 1000 mW. The required QoS target θ is 5 dB. The simulation is done two times. First, the Uncontrolled case, where every user is able to use his asked rate (as long as it is less than R_i^C) with constant power (500mW). Second, (GAME-PR), where GAME is applied with equal ω_P and ω_R to control power level and bit rate jointly.

Fig. 3 illustrates the effect of the increasing number of users on the average received E_b/N_o in the two cases. Although the average QoS is decreasing while adding more users, GAME-PR with joint power and rate control was all the time above the threshold θ. It is clear also that, on the average, only five users can be handled in the uncontrolled case while GAME coped with 19 users efficiently.

This success can also be seen in Fig. 4, which depicts the outage probability, another way of measuring QoS. This probability was measured for each mobile as the ratio of the time period that its E_b/N_o was below θ over the total call time. GAME-PR managed to decrease the outage probability considerably.

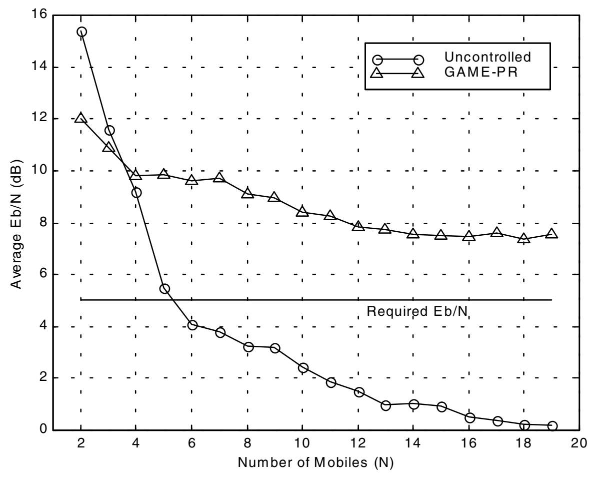


Fig. 3. Average received E_b/N_o (dB) Vs. Number of Users. 'o' represents the Uncontrolled case. 'Δ' represents joint Power and Rate control.

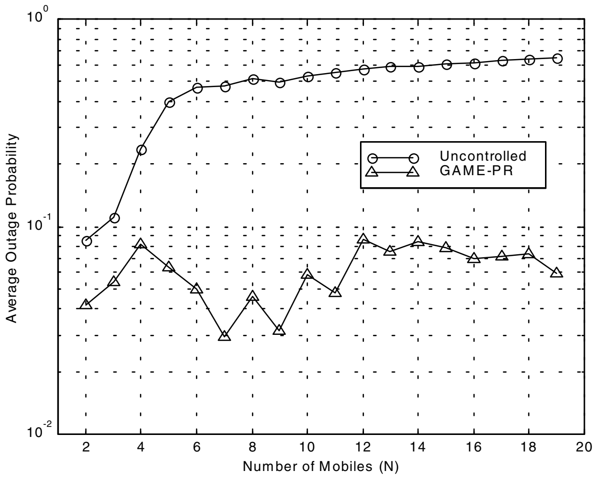


Fig. 4. Average Outage Probability Vs. Number of Users. 'o' represents the Uncontrolled case. 'Δ' represents joint Power and Rate control.

Preserving mobiles power was another objective achieved as shown in Fig. 5. Although the average power is increasing with the addition of new users, GAME managed to save at least 250 mW compared with the uncontrolled case.

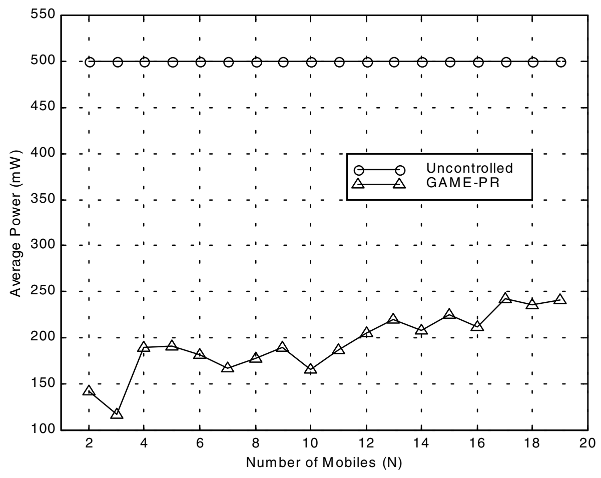


Fig. 5. Average Transmitting power (mWatt) Vs. Number of Users. ‘o’ represents the Uncontrolled case. ‘Δ’ represents joint Power and Rate control

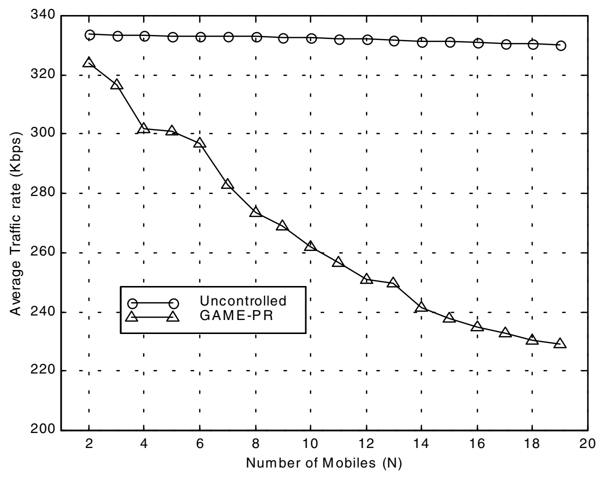


Fig. 6. Average Traffic bit rate (bps) Vs. Number of Users. ‘o’ represents the Uncontrolled case. ‘Δ’ represents joint Power and Rate control

Fig. 6 shows the price of this QoS enhancement and power conservation. It is some bit rate liberation. As indicated in (4), GAME tries to fulfill each user requested rate while maintaining the received signal quality (E_b/N_o) over the minimum value. This can be done through varying the users transmitter power levels as designated in (1).

Sometimes, it is impossible to satisfy a mobile high requested bit rate by simply increasing its power since this will increase the interference seen by other users significantly. Therefore the only solution is to cut the requested bit rate repetitively until the mating power is sufficient to top θ while introducing reasonable interference to others. This is the cause of the escalating difference between the requested bit rate (uncontrolled case) and the granted one (using GAME-PR) with the increasing mobiles number. Either using higher quantization factor or dropping frames can implement this rate-cut in MPEG traffic [4].

5 Concluding Remarks

The GAME scheme was introduced in this paper and applied to control video traffic. It includes the control of two main resources in a wireless network: mobile transmitting bit rate and corresponding power level. The main algorithm is to be implemented in the base station that forwards the controlling signals to the mobiles. The basic idea is that all the mobiles have to harmonize their rate and power according to their location, QoS, and density. The advantages of using genetic algorithms for optimization are numerous. Parallelism, GAME can be implemented as multiple synchronized threads to take advantage of the full processing power of the used hardware. Evolving nature, GAME can be stopped any moment while having the assurance that the current solution is better than all the previous ones. Scalability, mobiles can be added or removed simply by adjusting the chromosome length and leaving everything else intact.

The proposed scheme performed acceptably during the experiments done to test it. The enhancements over the uncontrolled case are substantial. The received signal quality (E_b/N_o) on the average has seen an increase of up to 7 dB and the outage probability is decreased tenfold. In the same time, the average power consumption is decreased by at least 50 %. These improvements help enhancing the current users QoS and extend the base station coverage to serve more mobiles. The noticeable GAME drawback is that it cut bit rates up to 30% of the requested rate. On the other hand, the comparison in bit rate between the uncontrolled case and the GAME-PR case, seen in Fig. 6, should stop at 5 users. The reason is that no more than 5 mobiles could be admitted to the system in the uncontrolled case as deduced from Fig. 3. Otherwise the QoS will fall below the required level. Therefore, this rate cut has contributed to the increase in the number of admitted users. In addition, the weight factor ω_R can be increased in order to give more importance to satisfying requested bit rate objective. Another problem is that the processing time to solve the optimization problem is seen to be proportional with the number of users.

Current research work is in progress to confront this shortcoming as well as applying the GAME on a wideband CDMA cellular network with mixed voice, data, video and web traffics while comparing with the current power-traffic control techniques.

References

1. CDMA Development Group. Service Description, User Requirements, and System Capabilities for Third Generation CDMA Systems Applicable to IMT-2000, *TR45.5*, Aug 1997
2. D. Goldberg, *Genetic Algorithms in Search, Optimization and Machine* Learning, Addison-Wesley, 1989
3. D. Kim, "Rate-Regulated Power Control for Supporting Flexible Transmission in Future CDMA Mobile Networks", *IEEE Journal on Selected Areas in Communications*, vol. 17, no. 5, pp. 968-977, 1999
4. D. Le Gall, "MPEG: A Video Compression Standard for Multimedia Applications", *Communications of the ACM*, pp. 47-58, April 1991
5. J. Lee and L. Miller, *CDMA Systems Engineering Handbook*, Artech House Publishers, Maryland, 1998
6. J. Zander, "Distributed Cochannel Interference Control in Cellular Radio Systems", *IEEE Transactions on Vehicular Technology*, vol. 41, no. 3, pp. 305-311, 1992
7. J. Zander, "Performance of Optimum Transmitter Power Control in Cellular Radio Systems", *IEEE Transactions on Vehicular Technology*, vol. 41, no. 1, pp. 57-62, 1992
8. J. Zander, "Radio Resource Management in Future Wireless Networks: Requirements and Limitations", *IEEE*, 1998
9. M. McTiffin *et al*., "Mobile Access to an ATM Network Using a CDMA Air Interface", *IEEE Journal on Selected Areas in Communications*, vol. 12, no. 6, pp. 900-908, 1994
10. O. Rose, "Statistical Properties of MPEG Video Traffic and Their Impact on Traffic Modeling in ATM Systems", Inst. Of Comp. Science, University of Wurzburg, Germany. Research Report No 101
11. S. Rappaport, *Wireless Communications*, Prentice Hall, New Jersey, 1996
12. S. Ulkus and R. Yates, "Stochastic Power Control for Cellular Radio Systems", *IEEE Transactions on Communications*, vol. 46, no. 6, pp. 784-798, 1998
13. Special Issue. IMT-2000: Standards Efforts of the ITU, *IEEE Personal Communications*, vol. 4, Aug. 1997
14. T. Hu and M. Liu, "A New Power Control Function for Multirate DS-CDMA Systems", *IEEE Transactions on Communications*, vol. 47, no. 6, pp. 896-904, 1999
15. U. Fiebig *et al*., "Design study for a CDMA-based third generation mobile radio system," *IEEE Journal on Selected Areas in Communications*, vol. 12, no. 5, pp. 733–43

PCP: An End-to-End Measurement-Based Call Admission Control for Real-Time Services over IP Networks

Giuseppe Bianchi[2], Flaminio Borgonovo[1], Antonio Capone[1], Luigi Fratta[1], and Chiara Petrioli[1]

[1] Politecnico di Milano, Italy
[2] Universitá di Palermo, Italy

Abstract. Distributed end-to-end measurement based connection admission control mechanisms have been recently proposed. The goal of these schemes is to provide tight QoS control on a per connection basis by means of measurements taken by the edge nodes and priority based forwarding procedure at internal nodes. Since the additional flows handling procedures are implemented at the border routers and the forwarding mechanisms are for flows aggregates only, the approach is fully scalable and compatible with the IETF Differentiated Service proposal. The aim of this paper is to propose specific schemes and to investigate the advantages and limits of the approach by analyzing the basic mechanisms and evaluating its performance. As a results, the paper shows that end-to-end measurements taken over a low priority probing packet stream is an effective and robust way to guarantee bandwidth and delay for real-time services characterized by fast traffic dynamics, such as Voice over IP.

1 Introduction

It is widely accepted that the best effort model of the today Internet is not able to satisfactorily support emerging services and market demands, such as IP Telephony. Real time services, in general, and IP telephony, in particular, require very stringent delay and loss requirements (150-200 ms end-to-end for toll quality voice), that have to be maintained for the entire call holding time. The analysis of the delay component in the path from source to destination shows that up to 100-150 ms can be spared for compression, packetization, jitter compensation, propagation delay, etc. [1,2], leaving no more than a few tens of milliseconds for queuing delay within the (many) routers on the path.

Many different proposals aimed at achieving such a tight QoS control on the Internet have been discussed in IETF. IntServ/ RSVP (Resource reSerVation Protocol) [3,4] provide end-to-end per-flow QoS by means of hop-by-hop resource reservation within the IP network. The price to be paid is a significant burden on the core routers, which are required to handle per flow signaling, to maintain per flow forwarding information on the control path, to perform per flow admission control, classification and scheduling. To overcome the scalability limits of this approach, several proposals (e.g. [5,6,7,8]) suggest to maintain only an aggregate QoS state (i.e. a run-time estimate of aggregate QoS parameters such as aggregate bandwidth and maximum delay) within the internal routers. The aggregate QoS state is then used by each router during set-up to verify whether enough resources are available to accept the new call without affecting the QoS of the accepted calls. Differentiated Services (DiffServ) enhancements to the Internet Protocol which have recently taken ground [9,10,11] represent an alternative attempt to provide a better than best effort service able to support real-time services while fully maintaining the stateless property of the current core IP routers. Diffserv distinguish between edge and core routers. While edge routers process packets on the basis of a finer traffic granularity (e.g. per-flow, per-organization), core routers only distinguish among a very limited number of traffic classes. Packets belonging to a given traffic class are identified by the bits in the DS field of the IP packet header, and are served by the routers according to a pre-defined per-hop-behavior. The result is that a variety of services can be constructed by a combination of: (i) setting DS bits at network boundaries, (ii) using those bits to determine how packets are forwarded by the core routers, and (iii) conditioning the marked packets at network boundaries in accordance with the requirements or rules of each service. While Diffserv easily enable resource provisioning performed on a management plane for permanent connections, it is not trivial to provide per-flow resource management and per-flow admission control, which seem to be necessary when tight per flow QoS is aimed at.

It is therefore important to understand to what extent an Internet stateless and scalable architecture, such as that envisioned by the Differentiated Services framework, has the capability to provide QoS comparable to that achievable by heavyweight (and ultimately abandoned) per-flow resource management approaches.

Recent proposals such as [12] envision a stateless Internet where packets carry control information from the edge to the core, and routers can take advantage of this information to improve QoS control. The opposite approach is taken in [13], where packet marking mechanisms within the core routers, along with pricing schemes, indirectly (i.e. by game theory) prevent new users from entering the network in case of congestion.

A more radical solution, named Phantom Circuit Protocol (PCP), apparently able to satisfy the stringent requirements of IP Telephony, has been envisioned in [14,15]. PCP is a fully distributed admission control scheme which relies on end-to-end in-band measures taken over an ad-hoc flow of probing packets to determine whether enough resources are available in the network to accept a new connection. Internal routers are completely stateless and are only required, compatibly and consistently with the DiffServ approach, to distinguish among probing and data traffics implementing a priority service discipline.

A solution similar in the basic principle, although targeted to a different environment (MPEG VBR), and employing different technological choices, has been proposed in [16]. More recently, although in a different framework, end-to-end probing has been considered in [13].

Scope of this paper is to give a comprehensive presentation of PCP, providing the rationale behind the proposed mechanisms. In particular, a new analytical model is introduced that is able to predict the effects of different measure-period lengths and allows to extend the performance evaluation to large capacity backbones that can not be simulated due to complexity.

The paper is organized as follows. In section 2, the basic idea of PCP is presented and discussed. Section 3 describes PCP when applied to a specific scenario of constant rate connections. Results for the more critical case of variable rate connections are reported in section 4. The analytical model, presented in section 5, allows to derive practical system limits and provides useful insights on the very-high-speed backbone scenario. Further remarks and conclusions are given in section 6.

2 The Phantom Circuit Protocol

The basic and characterizing idea, previously presented in [14,15], is to rely on end-to-end in-band measures over a Diff-Serv core network to determine whether enough resources are available in the network to accept a new connection.

In its simpler form, able to deal with real-time and best-effort traffic classes, PCP requires core routers to implement priority forwarding based on three priority classes, namely high priority for real-time traffic, medium priority for probing traffic, low priority for best-effort traffic.

In order to provide a minimum amount of bandwidth to best-effort traffic, bandwidth segregation mechanisms, such as Weighted Fair Queuing, can also be used, together with priorities. However, since the best- effort traffic does not influence the PCP operation it can be ignored and, in the following discussion, we assume only two priority levels.

Figure 1 shows the PCP setup scheme for a real-time connection. The setting-up source transmits a constant-rate signaling flow, composed of *Probing* packets tagged as low priority in the IP header. Upon reception of the first probing packet, the destination node starts measuring the probing packets arrival statistics over a given length period. A statistical test, which can be as simple as a comparison of the received bandwidth with a given threshold, is then performed to decide whether enough resources are available in the network to accept the considered connection. The decision is then notified back to the source, by means of one (or more) signaling packet, called *feedback* packet. Upon reception of an "accept" feedback packet, the source switches from probing to data phase, and starts transmitting *Data Packets*. Otherwise, if either a "reject" feedback packet is received or a time-out expires before receiving any feedback, the connection is terminated.

As it will be explained in the subsequent parts of the paper, the resources availability we have been referring to can always be expressed in term of bandwidth, say average, peak or effective bandwidth. Therefore, the probing and

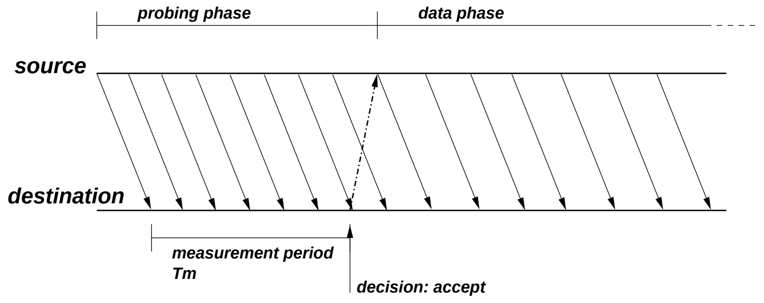


Fig. 1. PCP probing and data phases

measurement schemes discussed next aim at verifying the existence of enough bandwidth.

Assuming that the receiver exactly measures the bandwidth available in the network, a few conditions must be fulfilled in order to have PCP work properly. First, probing traffic must not contend the use of bandwidth against established traffic; otherwise, if the new requested bandwidth is not available, resources are subtracted to accepted connections, with a consequent QoS degradation. To avoid such QoS degradation, PCP tags the probing packets with lower priority than data packets. The core routers, DiffServ Compliant, handle packets in each output link with two distinct queues, one for data and one for probing packets, and serve a probing packet only when the data packets queue is empty. This forwarding mechanism guarantees that probing packets are forwarded only when bandwidth, unused by data traffic, is available. If the available bandwidth is not enough the probing flows will suffer severe quality degradation which will cause rejection of the call.

The second condition requires that the measure is able to capture not only the resources used by data packets but also the presence of concurrent probing connections. Otherwise, concurrent measures does not correctly "see" each other and the acceptance of multiple calls could lead to overload. This condition is obvious for requests of CBR flows of bandwidth B, and states that the probing flow must be a CBR flow with bandwidth not less than B.

Note that there is no need for an explicit bandwidth release mechanism since bandwidth is automatically released by ending packet transmission. This behavior leads to a third condition: the reserved bandwidth can not be temporarily released and the data phase must follow the probing phase without interrupting the flow. As far as the data phase is concerned, this condition is automatically satisfied in the case of CBR. In the case of VBR, however, it is requested that the flows maintain the "equivalent bandwidth" reserved at set up and therefore, when sources are inactive or the resources are under-utilized for long time, a conditioning mechanism, that send dummy packets according to the reserved traffic profile, must be provided. Such shaping procedures are common to resource reservation techniques based on traffic measurements (see for example

[5]) and constitute the price to be paid for the reduced complexity of the call admission procedure.

3 A PCP Scheme for CBR Traffic

In a scenario with only constant rate traffic, the delay, if the required bandwidth is available, is only due to the alignment of the different regular streams. This delay is bounded even if the aggregate load equals the link capacity and decreases as the multiplexing degree increases to the point that, in high speed links, it ceases to be a QoS concern. Such a behavior is shown in Figure 2, which reports different delay percentiles, measured in inter-arrival periods, in a ND/D/1 queue as a function of the link capacity when link utilization is 100%. Since we consider a scenario that refers to telephony applications in which a voice channel corresponds to 32 kb/s CBR, we see that links from 2 Mb/s upwards (more than 60 channels) can guarantee delays of a few milliseconds, in complete accordance with telephony delay requirements. Therefore, upon a new call request the call admission control must only ascertain if a 32 kb/s peak bandwidth is available in the network.

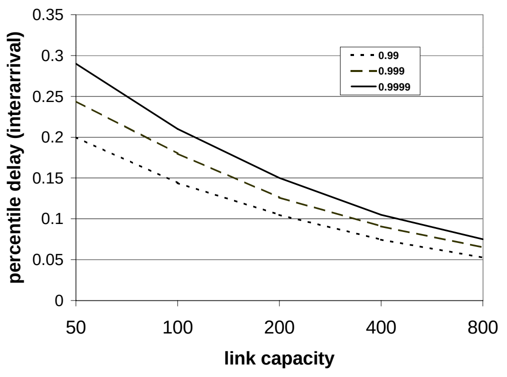


Fig. 2. N/N/D/1 delay percentiles versus offered load.

The receiver estimates the available bandwidth, B_a, by evaluating the mean, $M[X_i]$, of the interarrival times, X_i, of the probing packets. Due to jitter in the packet delay, the available bandwidth is estimated with an imprecision ΔB. To avoid the acceptance of a new call of bandwidth B when B_a is smaller than B it is sufficient to use a bandwidth threshold $B_t \geq B + \Delta B$ against which the measured bandwidth $B_m = 1/M[X_i]$ is compared. To avoid rejecting a call when the probing rate B_r is entirely available it is necessary that the probing packets are generated at a rate $B_r > B_t + \Delta B$.

With CBR traffic the measure duration must satisfy two conditions. The first one requires that the number of probing packets is sufficiently large to limit statistical fluctuations on B_m within the bound accounted for in B_t. The second one requires that the measure is long enough to capture the dynamic of the slowest traffic in the network. This implies that during the measure period a sufficiently large number of packets of the slowest connection are transmitted in the network.

In our simulations, assuming $B = 32$ kb/s and a packet size of 1000 bits, we have adopted values of $B_r = 1.2B$ and $B_t = 1.1B$. With link capacities of 2 and 20 Mbit/s, and $50,000$ connections per simulation, we have never observed misacceptance of a call even with a number of probing packets as low as 16, corresponding to 0.5 s measure duration.

In the following we show the simulation results for a single-link network, while results about CBR multi-link connections can be found in [14]. Figures 3 and 4 report the accepted load (throughput) versus the offered load, normalized with respect of the channel capacity, for 2 Mbit/s and 20 Mbit/s link capacity, respectively. The 99−percentile of the delay distribution expressed in ms is also shown for selected samples. Calls are generated according to a Poisson process and have an exponentially distributed duration with average 1, 3 and 10 min. The considered probing duration is $T_m = 1$ s, the probing bandwidth $B_r = 38.4$ Kbit/s and the bandwidth threshold $B_t = 35.2$ Kbit/s. To assure confident results, $50,000$ connections were simulated for each sample.

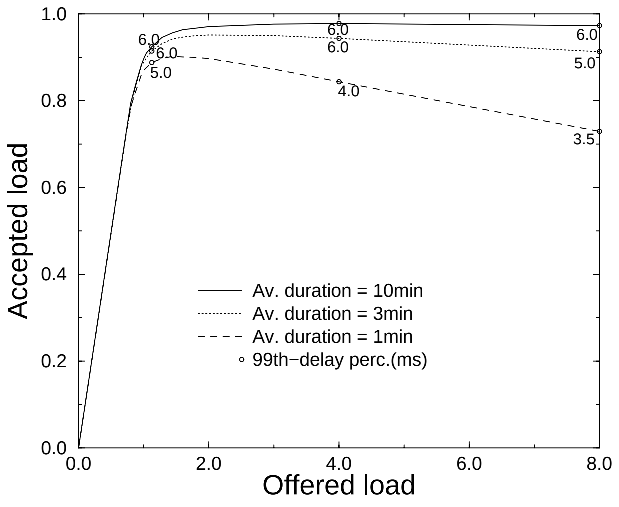


Fig. 3. Accepted load versus offered load for CBR connections, $C = 2$ Mbit/s

The offered traffic has been pushed up to eight times the link capacity in order to verify the PCP ability to control the admitted traffic even in extremely high overload conditions. All curves show that as the offered load increases, the accepted load grows up to a value that represents the channel utilization achievable by PCP. If the offered load is further increased, the reduction in

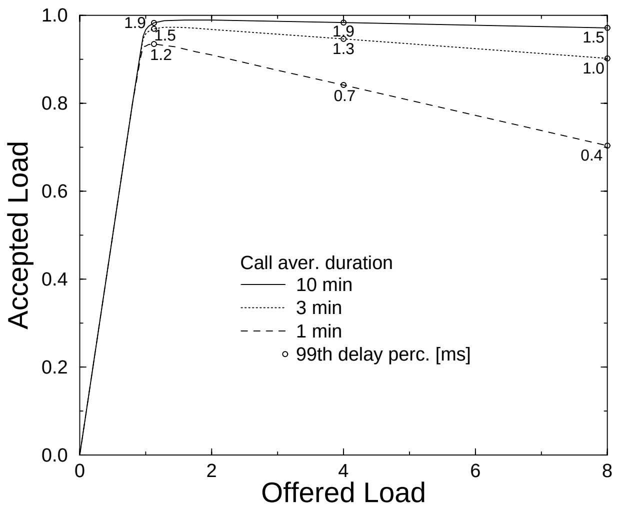


Fig. 4. Accepted load versus offered load for CBR connections, $C = 20$ Mbit/s

the accepted load, which is higher for shorter call duration, is due to contentions among the increased number of overlapping call set-ups that reduce the available bandwidth. This phenomenon, however, is far from being negative, since it helps protecting the admitted traffic under overload, whereas it has a negligible effect when the system operates in nominal conditions, i.e., with a offered load less than one.

The 99−percentile of the delay distribution, expressed in ms and shown for selected sample in the figures, has very low value even at the maximum accepted load. The simulation run with the probing duration $T_m = 0.5$ s, a probing rate $B_r = 1.1B$ and a threshold rate $B_t = 1.05B$ has produced similar results showing the robustness of PCP against the system parameters.

The performance of PCP for very high speed links can not be obtained by this simulation model due to the increased complexity. However, they can be evaluated using the PCP fluid model presented in [17] which shows very good matching with the simulation results.

4 PCP with Variable Rate Traffic

In the case of variable rate traffic the definition of a suitable bandwidth measure is more complicated than in the constant rate case addressed in the previous section. Unlike CBR, in VBR the aggregate traffic on a link varies in time according to source activities. The experienced delay depends on the average link utilization and grows without limit as the link utilization approaches 100%. Hence, to limit the delay, any effective call admission scheme must somehow exert a strict control on the link load.

In existing state-based CAC mechanisms (see for example [18,19]) load control is achieved by using ‘a priori’ traffic characterization (e.g. burstiness, peak and average bandwidth) to estimate the global equivalent bandwidth in use at a new call request. As the actual traffic can differ from its analytical characterization, these methods adopt estimations that are largely conservative, leading to under-utilization of resources. On the contrary, methods based on measures are intrinsically more precise. However, since PCP adopts an end-to-end measure, it can not directly observe the load on the intended network path and therefore must depend on a suitable probing-packets statistics to estimate the load only indirectly.

In the following we evaluate the following two mechanisms. The first mechanism, the Inter-arrival Average (IA), subdivides the measure period into sub-intervals and estimates the received bandwidth in each sub-interval. A new call is accepted if none of the estimates over the entire set-up period is below a pre-defined threshold. The rationale of this approach is to control the link load by controlling the ”peak bandwidth”.

The second method, the Peak Delay (PD), bases the statistical test on the queuing delays of probing packets and if at least one of them exceeds a pre-defined threshold the call is rejected. Also with this mechanism different values of the delay threshold correspond to different limits on the average load. A possible implementation of this technique requires that routers discard probing packets that exceed the pre-defined queuing delay. If only one probing packet is discarded the call is rejected. The implementation of this method would require changes in routers, which violates the PCP’s end-to-end approach. It has been considered here only to investigate whether a stricter control can improve performance.

To model the voice traffic with silence suppression, we have adopted the two state (ON/OFF) Brady model [20], where the ON (active - i.e. talkspurts) and OFF (silent) states are exponentially distributed with averages equal to 1 s and 1.35 s, respectively. The peak rate is 32 kB/s and the packet length is equal to 1000 bits, leading to a 31.25 ms packetization delay. We have simulated $2Mb/s$ and $20Mb/s$ single link scenarios in which calls are generated according to a Poisson process and have an exponentially distributed duration with average 1, 3 and 10 min. For reasons of space, we report in the following only the results obtained with 10 min. mean call duration, which appears among the three studied the most critical in terms of overload control.

Similarly to the CBR case, the probing-packets are generated at a rate 20% higher than the VBR-call peak rate. This leads to a nominal Inter-arrival time between probing packets of 26 ms. The measure period is fixed and equals to $0.5, 1$, and 2 s.

In all the presented results, the IA technique estimates the peak available bandwidth by averaging in each sub-interval and rejecting a call if at least one of such averages exceeds 27 ms. The size of the intervals depends on the number of inter-arrivals measured which has been taken equal to 5, 7 and 15 packets in our simulations, where, to assure confident results, 50000 connections were simulated for each sample. The accepted traffic versus the offered traffic is reported in Figures 5 and 6, for link capacity equals to 2 and 20 Mbit/s, respectively.

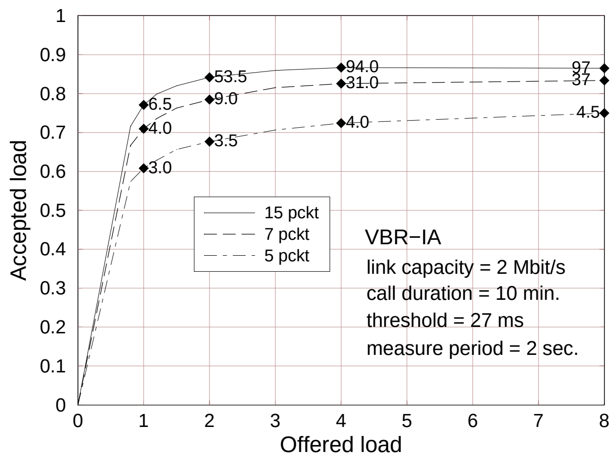


Fig. 5. *Accepted vs. Offered load for different test parameters in a 2Mb/s VBR-IA scenario.* 99− *delay percentiles are also reported for selected samples.*

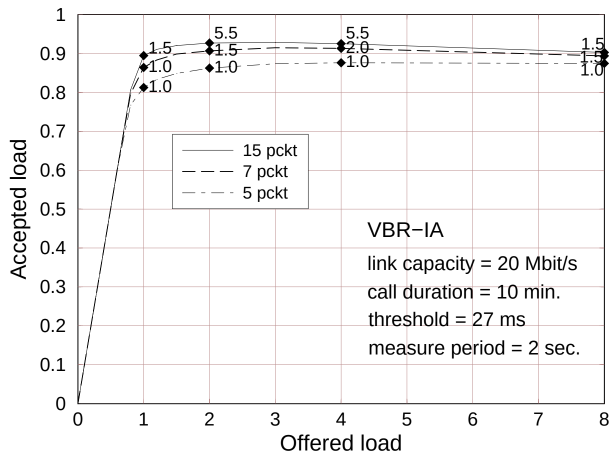


Fig. 6. *Accepted vs. Offered load for different test parameters in a 20Mb/s VBR-IA scenario.* 99− *delay percentiles are also reported for selected samples.*

The results obtained show, also in the case of VBR, that PCP exerts a powerful control even in very strong overload conditions. Furthermore, in all cases it can guarantee a 99 delay percentile in the order of few milliseconds, especially for high link capacity.

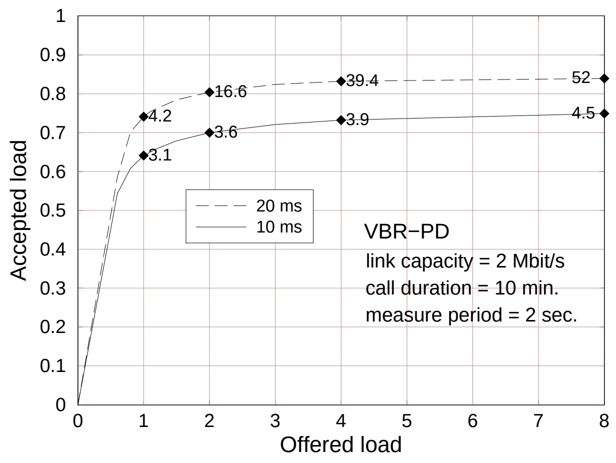


Fig. 7. *Accepted vs. Offered load for different test parameters in a 2Mb/s VBR-PD scenario.* 99− *delay percentiles are also reported for selected samples.*

Similar results has been obtained for the PD mechanism as shown in Figures 7 and 8. The delay threshold used for probing packets discarding has been set equal to 10 and 20 ms, showing that the smallest threshold tends to reduce the accepted load.

The comparison between IA and PD shows that the more strict control implemented in PD does not provide an appreciable improvement in results. In all the cases considered the parameters setting is not critical and the performance improve with the link capacity.

Several measure period lengths, ranging from 0.5 to 8 s, have been tested and the results are compared in Figure 9. As the measure period decreases the accepted load and the delay increase. To keep a small delay the acceptance threshold must be reduced.

As expected, we have observed that the delays achieved by PCP do not depend on the admission technique adopted, but on the accepted load only. This is graphically shown in Figure 10, which reports the 99th delay percentiles, in transmission time units (slot), as function of the accepted load for all the cases reported in the previous figures. The theoretical curve for the M/D/1 queuing system is also reported for comparisons.

In the operating conditions with very short delays, e.g., 1 ms for the 99 percentile, the transmission queue empties during any inter-arrival period (31.25 ms). For this reason, the process of packet arrivals in each period can be considered independent from period to period and, because of the many sources involved, equivalent to a Poisson process. As a consequence, the system behavior approximates the behavior of an M/D/1 queue. This is confirmed by Figure 10 that shows that, as long as delays are short, the experimental curves practically coincide with the M/D/1 curve. If the load is excessively increased the

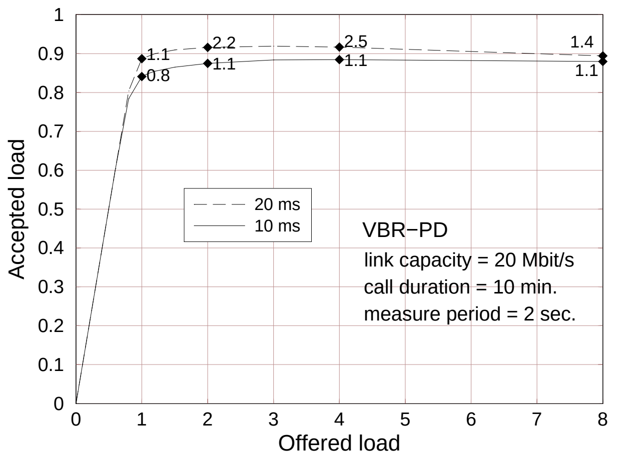


Fig. 8. *Accepted vs. Offered load for different test parameters in a 20Mb/s VBR-PD scenario. 99− delay percentiles are also reported for selected samples.*

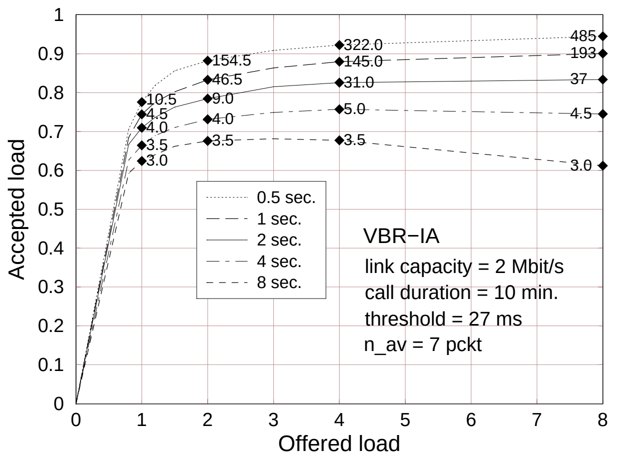


Fig. 9. *Accepted vs. Offered load for different measurement-period lengths in a 2Mb/s VBR-IA scenario. 99− delay percentiles are also reported for selected samples.*

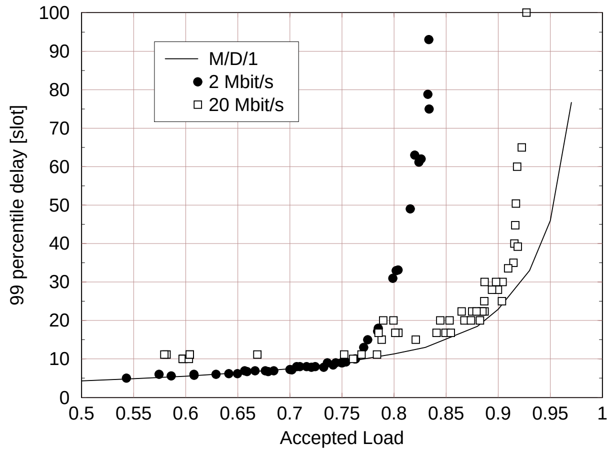


Fig. 10. 99% *delay percentiles, in transmission-time units, as function of the accepted load for the 2 and 20 Mb/s cases. The percentiles of total delay in M/D/1 queuing system is also reported.*

queue does not empty in each period and the delay sharply increases with respect to the M/D/1 case due to the periodic arrivals of packets in consecutive periods.

5 VBR Analytical Evaluations

The PCP mechanisms exposed in the previous section are too complex to be exactly analytically modeled. Nevertheless, analytical insides are needed to confirm the simulated results and to predict the behavior of larger systems. In this section we present an analytical model that, though rather simplified, is able to capture the PCP basic mechanism and to predict the influence of the measure-period length.

As we have already observed, both IA and PD mechanism try to capture the "peak bandwidth", i.e., the maximum value reached by the number of active sources $N(t)$. Therefore, in our model we refer to an "ideal" admission mechanism in which $N(t)$ can be directly observed, and assume that a new call is rejected if, during the probing period, $N(t)$ reaches a threshold A. In practice the value of A must be determined so that the accepted load can not exceed a predetermined value ρ^*, for a given overload.

The behavior of the "ideal" admission mechanism may be investigated by studying the Markov process $\{K(t), N(t)\}$ that represent the number of accepted sources and the number of active sources, respectively. The instantaneous load is directly related to the process $N(t)$ by the relation $\rho_i(t) = N(t)/C$. The average load (i.e. the quantity that needs to be controlled) is given by $\rho = E[N(t)]/C = \xi K(t)/C$, being $\xi = 0.4255$ the average activity of each voice source. According to the ideal mechanism, the call is rejected as soon as $N(t)$ reaches a threshold A.

The probability that $N(t)$ reaches the threshold A (absorption in A) during T can be expressed as

$$P_A = \sum_i \pi_i b_{iA}(T) \tag{1}$$

where $b_{iA}(T)$ represents the probability that, during T, $N(t)$ is absorbed into A starting from state i, and π_i is the steady state probability of the event $N(T) = i$.

We further assume that the process $K(t)$ is slow varying with respect to $N(t)$, so that, during the measure period, $N(t)$ behaves like the steady state process derived by the superposition of K on/off homogeneous markov sources. Owing to this assumption, probability (1) depends on $K(t)$ only through the value K of sources and, therefore $K(t)$ becomes a birth-death Markov process with "up" and "down" rates respectively given by:

$$\begin{cases} \lambda_K = \lambda(1 - P_A(K+1)) \\ \mu_K = K\mu \end{cases} \tag{2}$$

Though equations to get $b_{iA}(T)$, and therefore $P_A(K)$, can be analytically expressed [21], a close form solution is almost impossible to obtain and their numerical solution becomes impossible for the values of C we are interested. So, we resorted to a reduced Monte-Carlo method of evaluation, which provides directly $P_A(K)$.

The steady state solution π_i of process (2) provides the load as

$$\rho = \sum_K K\pi_K / C$$

To complete the mechanism, several values for A are tried in order to get an accepted load curve that does not exceed ρ^*.

Note that a limit of the model is that it does not account for the effect of concurrent access requests. However, this limit is conservative since concurrent access requests help reducing the admitted load in overload conditions.

In Figure 11 we report the analytical curves for the cases $C = 2$ Mb/s in which A has been chosen so that $\rho^*(8) = 0.75$, for different T. Similarly, Figure 12 reports the same curves for the case $C = 20$ Mb/s and $\rho^* = 0.9$. If we compare these results with those attained by simulations with both method IA and PD, we see that the ideal method is somewhat more effective in controlling the load, which is not surprising, since the real methods represent an attempt to approximate the ideal case. However, in accordance with simulation results, we note that the penalty associated with a reduction in the measure period is a slight reduction in the accepted load in nominal conditions, which reflects an increased difficulty in limiting the accepted traffic beyond this point. Fortunately, this effect is somewhat mitigated when the capacity C increases.

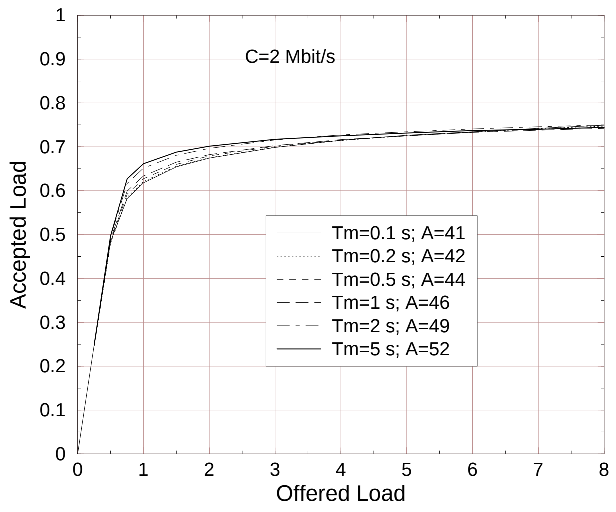


Fig. 11. *Accepted load versus offered load. Analytical derivation for the 2 MB/s case.*

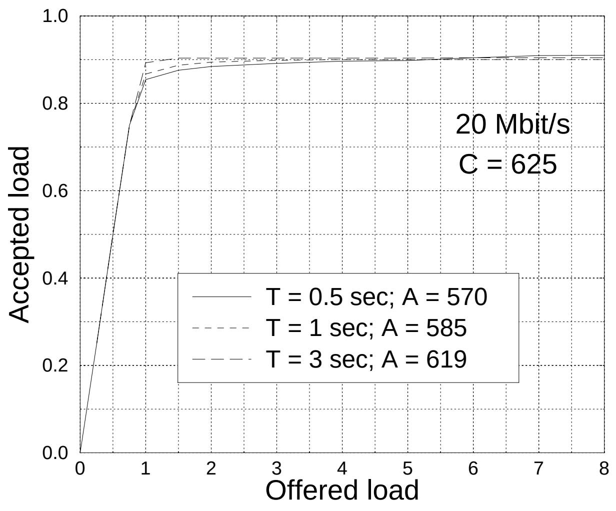


Fig. 12. *Accepted load versus offered load. Analytical derivation for the 20 MB/s case.*

6 Discussion and Conclusion

The results reported in this paper show that PCP is a mechanism able to provide guaranteed bandwidth to real-time CBR traffic, such as voice and video, on a call basis. The architecture is kept simple and scalable by shifting the complexity to the edges of the network. However, it is not free from limits, and further aspects remain to be investigated.

A drawback of PCP is that, differently from stateful centralized solutions, it provides only a single QoS degree. In fact, the measure mechanism based on two priority levels can not be consistently extended to give different priority levels for the accepted traffic. This means that heterogeneous real-time connections, with different loss/delay requirements, are forced to share the same priority and thus, regardless of how sophisticated the end-to-end measurement scheme might be, they ultimately encounter the same loss/delay performance.

Another issue that needs investigation is packet routing: more specifically, the effect of route changes on already accepted connections. When the routing changes, bandwidth might not be available on the new path, therefore causing congestion on the re-routed call and the calls already established on the new path. This problem arises from the actual way Internet packets are routed and affects all the QoS mechanisms alternative to IntServ/RSVP. It is addressed with techniques such as "Quality routing", [22,23,24] in which routing and re-routing is performed based on the bandwidth availability, and/or route pinning techniques. This approach requires the definition of a complete new routing mechanism and, again, forces core routers to be aware of the flows they are actually routing.

The solution with PCP is simpler, as it only requires a well engineered network and routers that know the bandwidth available on their links. Congestion, if no failure is encountered, is avoided by avoiding re-routing on already loaded links. This procedure, which is under study, is only internal to routers, does not cause interworking problems and is perfectly scalable.

Finally, the complete validation of the mechanism requires the performance evaluation in a multi-link scenario, which is in progress.

References

1. H. Schultzrinne, “Re-engineering the Telephone System” http://www.cs.columbia.edu
2. P. Goyal et al. “Integration of Call Signaling and Resource Management for IP Telephony”, IEEE Networks 13(3): 24–32.
3. R. Braden, L. Zhang, S. Berson, S. Herzog, S. Jamin, “Resource Re SerVation Protocol (RSVP)–Version 1 Functional Specification”, RFC2205, September 1997.
4. J. Worklawsky, “The use of RSVP with IETF Integrated Services”, RFC2210, September 1997.
5. W. Almesberger, T. Ferrari, J. Le Boudec, “Scalable Resource Reservation for the Internet”, IEEE IWQOS’98, Napa, CA, USA.

6. S. Jamin, P. Danzig, S. Shenker, L. Zhang, “A measurement-based Admission Control algorithm for Integrated Services Packet Networks”, IEEE/ACM Transactions on Networking, 5(1):56-70. Feb. 1997.
7. M.Grossglauser, D. Tse, “A Framework for Robust Measurement-Based Admission Control”, IEEE Trans. on Networking, vol. 7, no. 3, June 1999, pp. 293-309.
8. L. Breslau, S. Jamin, S. Shenken, “Comments on the Performance of Measurement-Based Admission Control Algorithms”, Proc. of IEEE INFOCOM 2000, Israel, March 2000.
9. K.Nichols, S. Blake, F. Baker, D. Black, “Definition of the Differentiated Services Field (DS Field) in the IPv4 and Ipv6 Headers”, RFC2474, December 1998.
10. S. Blake, D. Black, M. Carlson, E. Davies, Z. Wang, W. Weiss, “An Architecture for Differentiated Services”, RFC2475, December 1998.
11. Y. Bernet, J. Binder, S. Blake, M. Carlson, S. Keshav, E. Devies. B. Ohlman, D. Verma, Z. Wang, W. Weiss, “A Framework for Differentiated Services”, Internet Draft, February 1999.
12. I. Stoica, H. Zhang, ”Providing Guaranteed Services without Per Flow Management”, ACM SIGCOMM99, Boston, August 1999.
13. R. J. Gibbens, F. P. Kelly, ”Distributed Connection Acceptance Control for a Connectionless Network”, 16th International Teletraffic Conference, Edimburgh, June 1999.
14. F. Borgonovo , A. Capone , L. Fratta , M. Marchese , C. Petrioli, “PCP: A Bandwidth Guaranteed Transport Service for IP networks ”, IEEE ICC99, Vancouver, Canada, June,1999.
15. F. Borgonovo , A. Capone , L. Fratta , C. Petrioli ,“ VBR bandwidth guaranteed services over DiffServ Networks ”, IEEE RTAS Workshop, Vancouver, Canada, June, 1999 .
16. G. Karlsson, ”Providing Quality for Internet Video Services”, CNIT/IEEE 10th International Tyrrhenian Workshop on Digital Communications , Ischia, Italy, September 1998.
17. G. Bianchi, A. Capone, C. Petrioli, ”Throughput Analysis of End-to-End Measurement-Based Admission Control in IP”, to appear in INFOCOM 2000, Israel, March 2000.
18. R. Guerin, H. Ahmadi, M. Naghshineh, ”Equivalent capacity and its application to bandwidth allocation in high-speed networks”, IEEE Journal on Selected Areas in Communications, Volume: 9 Issue: 7 , Sept. 1991 , Page(s): 968 -981
19. J.A. Schormans, J. Pitts, K. Williams, L.G. Cuthbert, ”Equivalent capacity for on/off sources in ATM”, Electronics Letters , Volume: 30 Issue: 21 , 13 Oct. 1994 Page(s): 1740 -1741
20. Brady, P.T., “A Model for Generating On-Off Speach Patterns in Two-Way Conversation”, The Bell System Technical Journal, September 1969, pp.2445-2471.
21. E. Cinlar, Introduction to stochastic processes, Prentice Hall, 1975.
22. A. Orda, ”Routing with end to end QoS guarantees in broadband networks”, IEEE INFOCOM ’98, Volume: 1 , 1998 Page(s): 27 -34 vol.1
23. A. Dubrovsky, M. Gerla, S.S. Lee, D. Cavendish, ”Internet qos routing with IP telephony and TCP traffic”, IEEE ICC 2000 Volume: 3, Page(s): 1348 -1352
24. G. Apostolopoulos, R. Guerin, S. Kamat, A. Orda, S.K Tripathi, ”Intradomain QoS routing in IP networks: a feasibility and cost/benefit analysis”, IEEE Network , Volume: 13 Issue: 5 , Sept.-Oct. 1999, Page(s): 42 -54

Admission Control for Distribution of Smoothed Video Using Patching Algorithms

Gennaro Boggia, Pietro Camarda, Maurizio Tortorici

Politecnico di Bari – Dip. di Elettrotecnica ed Elettronica
Via. E. Orabona n.4 – 70125 Bari (Italy)
boggia@deemail.poliba.it; camarda@poliba.it

Abstract. The resource sharing techniques in Multimedia on demand systems allow the simultaneous service of a large number of requests with considerable savings in terms of network bandwidth and server resources. In this paper, we report the results of a study that analyzes several key aspects of video distribution systems, in the hypothesis of exploiting patching techniques. Synthetically, such a technique consists in serving a client request by an existing stream for that video object (if there is any) which is buffered in the client equipment and simultaneously a new stream is requested to the server for the frames already transmitted. For such systems, a performance analysis has been carried out, exploiting analytical and simulation models. The results show the main system performance (efficiency, aggregate bandwidth, etc.) allowing to decide about the acceptance of a new service request based on QoS criteria.

1. Introduction

A challenging issue in future telecommunication networks will be the Quality-of-Service (QoS) support for real time distribution of video streams. In such systems, which form the core of applications like Video on Demand (VoD), distance learning, internet video broadcast, etc, many approaches have been studied to improve their efficiency by managing critical resources, such as network bandwidth [1,2]. The target is to reduce the bandwidth demand, increasing the number of client requests which can be served simultaneously by multicasting the same video stream and thus reducing the aggregate bandwidth of the overall video traffic. The most important techniques proposed to implement the mentioned optimization comprehend various approaches that can be exploited simultaneously; we have: batching of requests [3-4], merging of video streams [5,6], buffering in RAM memory [7,8] and patching schemes [9,10].

We suppose that, to reduce rate variability, smoothing algorithms are exploited in transmitting video streams [11,12]. These algorithms, exploiting client buffer, can be employed for optimizing network resources utilization. The technique utilized to reach this purpose consists in transmitting as long as possible pieces of the same film with a constant bit rate, that varies from piece to piece. This can be done only if we suppose that there is a buffer used to store data during the transmission of the film, and another one used to store the received data before sending them to the MPEG decoder.

The smoothing algorithm proposed in [12] determines the longest segments of the film that can be transmitted in a CBR manner, minimizing the bit rate variability, and, at the same time, preventing buffer overflows and underflows.

We will use patching algorithms to make the multicast possible. Those schemes are more efficient then batching ones, since they avoid the startup delay of the latter. However, existing patching schemes are not practical and offer some problems, such as an intrinsic incompatibility with smoothing and the use of a variable size buffer in client's memory. That is the reason why, in this paper, we propose more practical algorithms that, besides being compatible with smoothing techniques, are more efficient in the use of memory then the existing ones. Subsequently, we evaluate their efficiency, and develop a statistical admission control scheme based on the estimation of aggregate bandwidth needed by a given number of video streams that are being transmitted following the patching algorithms. In particular, in section 2, after summarizing existing patching schemes and related issues, we present our patching algorithms and performance analysis. In section 3, we present an admission control scheme for single-trace stream, comparing performance results with existing algorithms and with simulations. The conclusions, in section 4 close the paper.

2. Patching Algorithms

Patching schemes require that clients have a *workahead buffer* in which they can store part of the video directly from the ongoing transmission, and sufficient bandwidth to simultaneously get two streams (or even more, in some patching schemes, as shown in [9]).

2.1 Existing Patching Algorithms

To summarize existing patching schemes, we consider the Periodic Buffer Reuse (PBR) patching [9] (disregarding the too complex GBR one), (see Figure 1) and we refer to the last complete transmission of the film from the server as to the *Most Recent Complete Transmission* (MRCT) (the upper line in the figure). We assume a discrete-time model, with a granularity of one frame and suppose that the video length is of N frames (from 0 to N-1). Moreover, we suppose that the i-th client's request from the beginning of the MRCT (dedicated to the first one) arrives at time t_i (assuming that frame 0 of MRCT is sent at time 0).

Therefore, the server must transmit all frames between 0 and t_i directly to the client, while the latter retrieves frames from t_i to t_i+B from the ongoing transmission, where B is client's buffer size expressed in frames (i.e. variable size in bytes, since video flows are usually VBR). If B>t_i, there is no need to continue transmitting from the server directly to the client, because when client playback exceed t_i, he starts emptying the buffer B, thus allowing to store next frames in the same buffer, retrieving them from the MRCT. If B≤t_i, as shown in figure 1, frames from t_i+B to 2t_i cannot be taken from the ongoing transmission, and must be read directly from the server just before their playback time (*deadline*). At time 2t_i the client start emptying buffer and can simultaneously read frames from the MRCT, until time $2t_i+B$, when frames previously contained in buffer are over, and he must read frames directly from server. The process begins again, and repeats in a periodic fashion, as shown in figure. This periodic behavior gives the name to this algorithm. The optimization in network load is a time-average reduction of the bandwidth requirement in the common path of the multicast, since the server does not send another complete transmission to the client. Moreover, when the client does not need to read the MRCT, in a well-organized network we can suppress the MRCT itself in the part of the path dedicated to the that client within the multicast resources, also reducing the receiver load. Besides, we avoid the startup latency of batching schemes [3, 4, 9, 10].

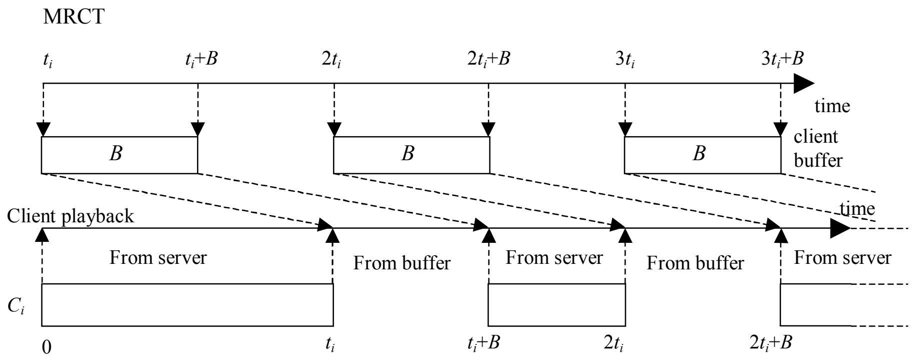


Fig. 1. Graphic description of PBR patching.

It is obvious that if t_i>>B, the PBR patching is not useful. Therefore, all patching schemes include a threshold T. The server starts a new complete transmission (that becomes the MRCT) whenever the arrival instant exceed this threshold value T; in other words, if $t_i>T$ the server will transmit the entire video to client i, which becomes client 1 for the new repetition of the process, and applies patching to the next clients with this new MRCT. When a client arrives T frames later of the beginning of this complete transmission, the process will restart again. The reason of the threshold is the following: the patching efficiency reduces with the increasing of the delay, computed from the beginning of the MRCT, with which a client requires the film. Therefore, there is a time in which starts to be worthwhile to retransmit the whole video and to apply patching to this, instead to the previous MRCT, because it is more efficient. We can quantify this concept by estimating the average number of frames sent to a client, say $E[W_c]$, where W_c represents the random variable indicating the number of frames sent to a client. As shown in [9], by estimating this number of frames in function of the threshold value T, we can find the optimal threshold value T_{OPT}, simply minimizing $E[W_c]$.

Since we have a renewal process, with renewal points represented by the beginning of each complete transmission (see [9] for details), we can restrict our analysis to the first complete transmission and flows patched to it in deriving an expression for $E[W_c]$. We denote with W and A the random variables indicating the total number of frames sent to all clients arrived within a renewal interval and the number of these clients, respectively. If λ is the average number of clients per time unit, we have [9]

$$E[W_c]=E[W]/E[A]; \qquad E[A]=1+\lambda T; \qquad E[W]=N+p\sum_{t=1}^{T}G(t) \qquad \textbf{(1)}$$

where p is the probability of at least one arrival within a unit of time (we can batch together more requests that occurs within a frame period or within a small group of consecutive periods: batching and patching can be applied together). $G(t)$ is the total number of frames sent to a client whose request arrives at time t.

A key issue of PBR patching is that it requires a variable buffer size in memory. It is easy to see that both server and client cannot compute the buffer requirement, be it in frames or in bytes, even if we refer to its peak occupancy, without involving a severe process of negotiation and repeated computation of the schedule that makes PBR patching unusable.

Besides, for smoothed traces the server sends B frames not in B frame periods, but in a variable and non-integer number of them. Therefore, we should apply PBR patching in a continuous time form, and not in the discrete one we have described above. In the sequel, we propose some schemes that overcome these problems, since they refer to the byte schedule of the flow to be retransmitted.

2.2 New Patching Algorithms

The fundamental hypothesis we make is that buffer size is fixed in bytes. This avoids the ambiguity in the determination of its size (indeed, the client will give the size of the buffer to the server and the latter computes the schedule according to our algorithm only once). Besides, it partially helps us in reconciling patching with smoothing. However, both patching and smoothing requires a buffer, but these buffers must be distinct, because the client reads both the ongoing transmission, putting it in the patching buffer (of size B_i for the i-th client), and the server transmission dedicated to him, putting it in the smoothing buffer. Another fundamental hypothesis is that smoothing buffers of all clients are of the same size, say b, allowing the same schedule for all clients.

SPS Algorithm. Now we introduce the first and simplest patching algorithm, that we will call *Simple Patching-Smoothing* (SPS) algorithm. In the sequel of the paper, we will refer to the i-th client as the batch of clients who made request for the same film within a frame period. The server sends to the client the data contained in the first t_0 intervals (to which we will refer simply as intervals), where t_0 is the interval in witch falls the client request. In the meantime, the client itself can read the intervals from t_0+1 to t_1 (wich is the last interval that can be stored in buffer B_i without causing its overflow) and put them in the patching buffer. After reading from C_i the first t_0 intervals (where C_i denotes the virtual channel dedicated to the i-th client), the client starts emptying B_i and does not start again to fill it up until it is completely empty (at time t_0+t_1+1, see Fig. 2). Let $\aleph$ be the set of natural numbers, $a(t)$ the amount of bytes sent in interval t (i.e. the transmission schedule), and $A(t)=\sum_{i=1}^{t} a(i)$. Therefore, we have

$$t_1=\min\{t\in[t_0+1,N]\cap\aleph|A(t)-A(t_0)>B_i\}-1 \qquad \textbf{(2)}$$

where $A(t)-A(t_0)=\sum_{i=t_0+1}^{t} a(i)$ is the total number of bytes read in $[t_0+1,t+1[$.

In instant t_0+t_1+1 patching buffer is totally emptied, so the client starts to read again from C_0 and, in the meanwhile, from C_i. This can be performed until interval t_2, which is defined as

$$t_2=\min\{t\in[t+t_1,N]\cap\aleph|A(t)-A(t+t_1-1)>B_i\}-1\,. \qquad \textbf{(3)}$$

We can iterate this process until we determine that instant t_q such as $t_q+t+1>N$, which is the condition that determines the end of the cycle (see [15] for details).

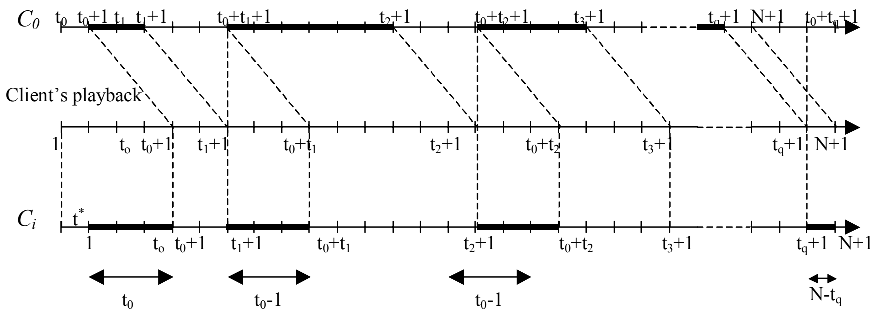


Fig. 2. Explanatory scheme of SPS algorithm. Bold lines indicate the source of the intervals that are then recomposed at the center. Data taken by C_0 are stored in buffer B_i, while those taken by C_i, after that they have been stored in buffer b, are directly played.

TBR Algorithm. The simple SPS scheme is able to conceal patching with smoothing, but it suffers some inefficiency due to the mechanism exploited for buffer reuse. At this point we present the basic principles for *Total Buffer Reuse* (TBR) patching, the most efficient (although the most complex) of our patching schemes, that overcome the inefficiency of SPS.

We start exploiting a single time axis, fixing the instant t=1 as the beginning of the transmission dedicated to the client and referring the MRCT axis to this system. Hence, the transmission schedule for the client remains $a(t)$, while the transmission schedule on C_0 becomes $a_0(t)=a(t+t_0)$. The patching buffer begins to collect data from C_0 from instant 1. We need a boolean vector **CS** of length N and a boolean matrix **RS** of size 2xN. If CS_j=TRUE (j = 1,2…N) then at time j the server sends $a(j)$ data on C_i, otherwise he sends nothing. If $RS_{j,1}$=TRUE ($RS_{j,2}$= TRUE) then in interval

j the client must read from C_0 (C_i), otherwise not. To compute **CS** we need to know only the buffer occupancy in instant t-1, which can be computed from the knowledge of $a(t)$ and $a_0(t)$ and stored in a scalar, say X. From the knowledge of X at time t, of $a(t)$ and of CS$_t$, we can compute **RS** (see [15] for details). The latter must also be sent to the client, to allow him to know whether he should read from C_0 or from C_i. The complexity of this algorithm is $O(N)$, but it exploits client buffer every time it can be filled, hence its name.

OSPS Algorithms. The great limit of the SPS algorithm is that it does not fill client's buffer unless it is empty. The TBR patching does it, but at a great complexity price. Then, we modify the SPS patching in order to fill the buffer before it is empty without doing to much calculations, as the second scheme does. Therefore, we must estimate the interval in which the client may resume to fill his buffer without incurring in overflow. The simplest evaluation, assumes a CBR transmission with rate r, the peak rate of the real transmission schedule, between instants $t_{i-1}+t_i$ and $t_{i+1}+1$ (see Fig. 2); we use this transmission to evaluate if the remaining buffer space is sufficient to store the remaining intervals, say x, which requires rx bytes. We call this new algorithm *Optimized SPS* (OSPS).

The previous basic scheme can be further improved. As a matter of facts, we propose three variants, in which we sharpen the estimation of the instants in which to resume to fill the buffer, paying in complexity. They are, respectively: the evaluation of the peak rate locally between $t_{i-1}+t_i$ and $t_{i+1}+1$ (OSPS1); the evaluation of this peak rate only for the remaining intervals we should send in advance compared to the normal SPS schedule (OSPS2); the computation of the effective amount of data that we should send in those remaining intervals (OSPS3). Those schemes are sorted in increasing complexity order. The choice of one of those algorithms is a compromise between their load on the specific machine on which the designer works and the performance that should be reached; the latter depends also on client's buffer size and arrival time. Before comparing these performances, we put our interest on the computation of the threshold for the SPS and the TBR algorithms.

2.3 Threshold Issue

As regards our patching schemes, we first change the units of the (1) using intervals instead of frames and then compute $G(t)$ depending on the algorithm used. Note that we will continue to use the same unit of time used till now (i.e. frames).

For the SPS algorithm, we can consider three cases, with reference to Fig. 2:

- If $t_1 \geq N$ then the client's buffer is large enough to store all the remaining data of the film, hence $G(t)=t$.
- If $t_I<N$ and $t_q \leq N$, $G(t)$ is the sum of three terms: the first is given by t, of course, the second is given by the q-1 intervals in Fig. 2, each of length t-1, and the latter is made up of the last intervals the server must transmit. Hence $G(t)=t+(q-1)(t-1)+N-t_q$.
- If $t_I<N$ and $t_q>N$, $G(t)$ is the same of the previous case, but without the last term.

Finally, we have

$$G(t)=\begin{cases} t & \text{if } \ t_1 \geq N \\ t+(q-1)(t-1)+N-t_q & \text{if } \ t_1 < N \ \text{ and } \ t_q \leq N \\ t+(q-1)(t-1) & \text{if } \ t_1 < N \ \text{ and } \ t_q > N \end{cases}. \tag{4}$$

For a fixed buffer size, we have to compute $G(t)$ for t=1,2…T, using the SPS algorithm to determine t_1, q and t_q. Then we can obtain $E[W_c]$ from (1). This procedure must be repeated for every threshold value T. Therefore, we have not an analytic expression to minimize to determine T_{OPT}.

To find a simpler method, we approximate the schedule $a(t)$ with its mean $\bar{a}$ and compute $G(t)$ consequently. First, we disregard the term N-t_q in the second of the (4), thus avoiding to compute t_q. Since we begin to fill the buffer only when it is empty, the mean number of intervals in which we do not transmit on the C_i channel is $\alpha=\lfloor B/\bar{a} \rfloor$[1]. Consequently, q-1 is given by the ratio between the remaining number of intervals, i.e. N-t, and the number of intervals of which is made a single transmission-no transmission group, i.e., α+t−1. Hence:

$$q-1 \approx \left(N-t_q\right)/\left(\alpha+t-1\right). \tag{5}$$

Moreover, the condition of the first of the (4) becomes $t \geq N-\alpha$. If $T<N-\alpha$, then $t \geq N-\alpha$ is never true; therefore $G(t)$ reduces to the third of the (4). Hence:

$$\begin{aligned} E[W]&=N+p\sum_{t=1}^{T}G(t)\approx N+p\sum_{t=1}^{T}\left(\frac{N-t}{t-1+\alpha}(t-1)+t\right)=N+p\sum_{t=1}^{T}\frac{Nt+\alpha t-N}{t-1+\alpha}= \\ &=N+p\left((N+\alpha)\sum_{t=0}^{T-1}\frac{t+1+\alpha-\alpha}{t+\alpha}-N\sum_{t=0}^{T-1}\frac{1}{t+\alpha}\right)\cong N+p\left[(N+\alpha)\left(T+(1-\alpha)\ln\frac{T-1+\alpha}{\alpha}\right)+\right. \\ &\left.-N\ln\frac{T-1+\alpha}{\alpha}\right]=N+p\left[(N+\alpha)T+\alpha(1-\alpha-N)\ln\frac{T-1+\alpha}{\alpha}\right] \end{aligned} \tag{6}$$

where the last approximation stems from $\sum_{t=0}^{T-1}\frac{1}{t+\alpha}\cong\int_0^{T-1}\frac{dt}{t+\alpha}$.

1) If $T \geq N-\alpha$ we must dived the summation in two terms:

$$E[W]=N+p\sum_{t=1}^{N-\alpha-1}G(t)+p\sum_{t=N-\alpha}^{T}G(t)=N+p\sum_{t=1}^{N-\alpha-1}\left(\frac{N-t}{t-1+\alpha}(t-1)+t\right)+p\sum_{t=N-\alpha}^{T}t\,. \tag{7}$$

The first summation is exactly the previous one, simply substituting $N-\alpha-1$ at T, while the latter corresponds to $(T-N+\alpha+1)(T-N+\alpha+2)/2$.

Therefore, we have three possibilities: to use the exact computation of $G(t)$ applying patching at every time t, to use the summation which stems from the

[1] In the following, we assume that the clients have the same patching buffer B. Moreover, the assumption of a CBR transmission justifies the use of intervals instead of the amount of data, since this is, on average, proportional to the number of intervals.

approximation of $a(t)$ with $\overline{a}$ (to which we refer as the first approximation), or to use the analytic approximation of the summation (to which we refer as to the second approximation).

In Fig. 3, we show the comparison between the results produced by these three methods.

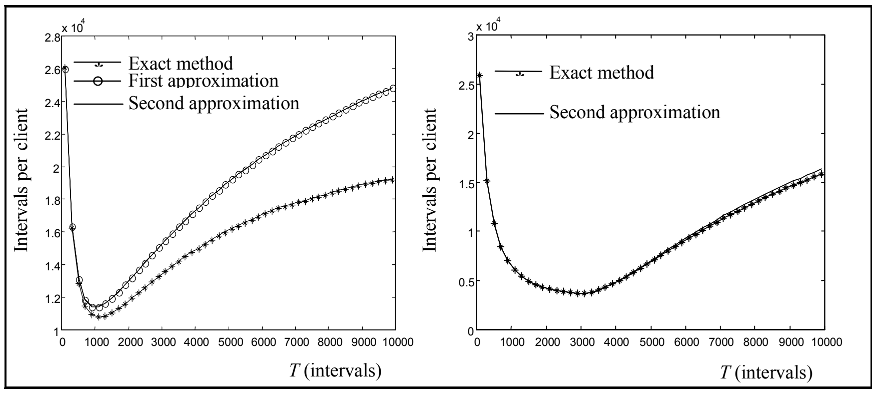


Fig. 3. For this simulation we use the first 40000 frames of the film "Asterix", with b=4 MB, B=8 MB, λ=10 clients/min., Poisson arrival distribution. On the left, we can see the results for SPS patching, while on the right we show the results for the TBR one.

The approximations are better for little T. The difference between the first and second approximation is almost imperceptible, while the threshold value analytically computed differs of about the 8% from the one computed with the exact method, an acceptable approximation, if we consider the advantages coming from the analytic expression. In other simulations we obtained errors slightly inferior to this[2].

Now we put our interest in the computation of the threshold of the TBR algorithm. Again, we assume a CBR transmission of value $\overline{a}$. TBR patching uses client's buffer every time it is possible. Therefore, during the first t intervals the client simultaneously reads α intervals from the MRCT. In the next t intervals, the client plays the α intervals read before and simultaneously reads other α intervals from the MRCT, while in the next t-α periods gets data from the server. This repeats in a periodic way, hence on average TBR patching behaves as PBR patching with buffer size α. Therefore, we can obtain the same performance of PBR patching with a buffer of size α, instead of a buffer size allocated on the peak occupancy. In conclusion, we can apply the results obtained in [9] to determine $E[W_c]$ for TBR patching. The results are shown in Fig. 3. We note that, due to a better efficiency of

[2] Note that the threshold value is very low. The reason is that we have used a relatively small patching buffer. For instance, in [9], a good threshold value is obtained with a buffer of an average of 225MB.

TBR, the value of the threshold is shifted on the right when compared with the previous case.

2.4 Performance Analysis

Now we compare the efficiencies of the previous algorithms, where the efficiency is defined as

$$1-\frac{\text{total amount of data sent on} C_i \text{ (bytes)}}{\text{film size (bytes)}}=1-\left(\sum_{\{t|\text{server transmits over} C_i\}} a(t) \Big/ \sum_{t=1}^{N} a(t)\right). \qquad \textbf{(8)}$$

In Fig. 4 we show the results of our simulations with the same settings but with different buffers. In both figures, we can see how SPS and TBR schemes are, respectively, the least and the most efficient, (this is a constant throughout all the simulations we have performed); moreover, only the OSPS3 algorithm distinguishes itself from the others.

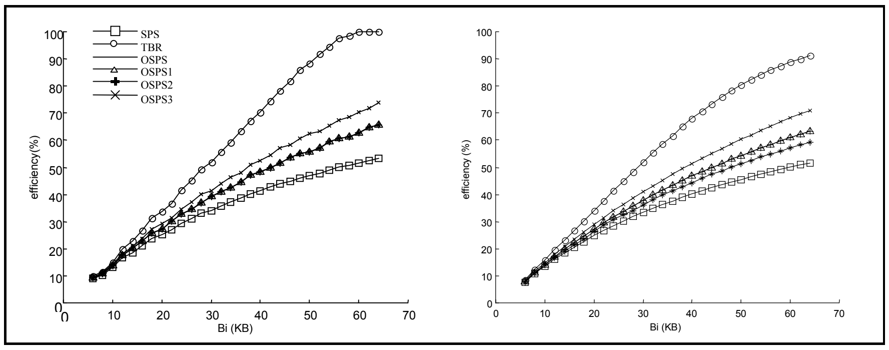


Fig. 4. Comparison between the various algorithms with medium buffer size: arrival time t_0=20 (frames, as in the whole sequel), film "Asterix"[3], smoothing buffer b=4 MB (on the left) or b=64KB (on the right).

The most important aspect we note is the reduction of the efficiency with the reduction of the smoothing buffer. We have observed this behavior in all the tests we have made. Then we made new particular tests (using a stationary source with different distributions and variances) that confirm the following rule: reducing the variability (i.e., the variance) of the source and/or its minimum and peak rates, will generally result in an increase of efficiency of the patching schemes (we do not report those simulations because of lack of space). This is the reason why we use the optimal smoothing presented in [11].

[3] All clips used are of 40000 frames, if not differently specified.

The efficiency depends not only on the buffer size, but also, and above all, on the arrival time of the client. In Figure 5, on the left, we compare the algorithms for t_0=8000; we can see how all the schemes are equivalent and almost useless.

We can obtain the same results with t_0=20 but with little buffer sizes. If we do not consider the threshold, we can say that in those events the best patching algorithm is the SPS.

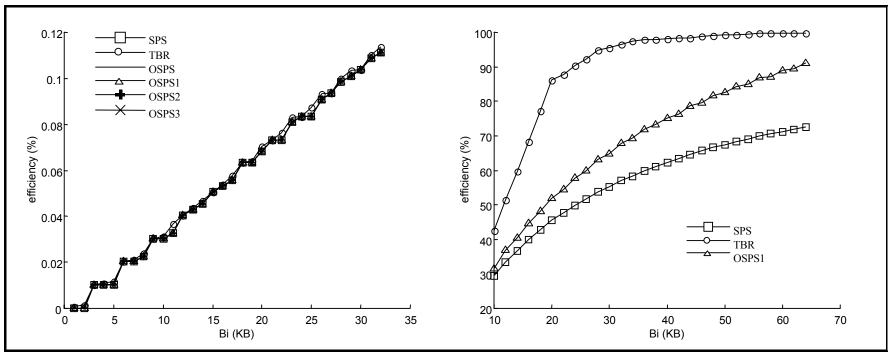


Fig. 5. Left: comparison of the algorithms in the case of t_0=8000, b=4 MB, film of the series "007". Right: saturation of the efficiency, film "Star Wars", smoothing buffer b=4 MB, t_0=20.

In Figure 5, on the right, we compare only three algorithms. We note how the TBR patching efficiency "saturates", i.e. reaches a value that cannot be exceeded. Actually, there is a certain buffer size beyond which it is useless to go, since the algorithm considered reaches the maximum theoretic efficiency. This is not one, because the first t_0 intervals must be transmitted in any case, but is $(N-t_0)/N$. This figure brings us to another conclusion: for great buffer sizes it is unnecessary to use the TBR patching.

3. Single Source Call Admission Control (CAC)

In this section, we present a CAC algorithm that exploits the correlation between the MRCT and patched streams. Actually, the patched sources generate streams that are null or exactly equal to the MRCT, within a frame period, with the only difference that they are time shifted. In the first subsection we present a general overview of the Chernoff bound based CAC algorithm and related issues, while, in the second subsection, we modify it for the setting considered in this paper.

3.1 A Chernoff Bound Based CAC

Suppose we have I different sources and J_i flows for the source of type i=1,2,…I. Let c be the channel capacity. Let $a_{ij}(t)$ the amount of traffic generated by source j of type i at time t. We assume that $a_{ij}(t)$ has a stationary distribution (then $a_{ij}(t)$=a_{ij} for all t).

We assume that a_{ij} is represented by a k_i-state random variable that assumes the values $r_1^{(i)} \le r_2^{(i)} \le \ldots \le r_{k_i}^{(i)}$ (the transmission rates) with probabilities $p_q^{(i)}$ (q=1,2…k_i). The total amount of traffic is a random variable a given by the sum of all the random variables representing the sources: $a = \sum_{i=1}^{I} \sum_{j=1}^{J} a_{ij}$. In the hypothesis of independent sources and c to approach infinity, keeping the ratio J_i/c constant, the loss probability can be estimated by a refined expression of the Chernoff bound:

$$P(a \ge c) \approx \frac{1}{\theta^* \sqrt{2\pi \Lambda''(\theta^*)}} e^{-\theta^* c + \Lambda(\theta^*)} \qquad \textbf{(9)}$$

where $\Lambda^*(\mu) = \sup_{\theta \ge 0} \{\theta\mu - \Lambda(\theta)\}$, $\Lambda(\theta) = \sum_{i=1}^{I} J_i \log M_i(\theta)$ and $M_i(\theta) = \sum_{q=1}^{k_i} p_q^{(i)} e^{\theta r_q^{(i)}}$ is the moment generating function of a_{ij}; θ^* is the solution of the equation $\Lambda'(\theta) = c$ ($\Lambda'(\theta)$ and $\Lambda''(\theta)$ are the first and second derivatives of $\Lambda(\theta)$, respectively).

Since we know the desired loss probability λ, we can apply the Chernoff bound as follows: substituting c with $\Lambda'(\theta)$ and P($a \ge c$) with λ in (9), we obtain θ^* solving this equation, that we can write in another form:

$$\log\lambda = \Lambda(\theta) - \theta\Lambda'(\theta) - \log\theta - \frac{1}{2}\log\Lambda''(\theta) - \frac{1}{2}\log(2\pi) \ . \qquad \textbf{(10)}$$

Then, we deduce the bandwidth requirement c^* to satisfy the given loss probability from:

$$c^* = \Lambda'(\theta^*) = \sum_{i=1}^{I} J_i \frac{M_i'(\theta^*)}{M_i(\theta^*)} \ . \qquad \textbf{(11)}$$

Let us see how operates a CAC algorithm based on the Chernoff bound. Given the loss probability λ and the channel capacity c, suppose that a new call of type q ($1 \le q \le I$) arrives. First we compute θ^* from (10) and then c^* from (11), replacing J_q with J_q+1. Finally, we admit the new flow if $c^* \le c$.

The computational cost of this algorithm lies on the computation of the moment generating functions of the I sources when solving (10) (this can be easily done with Newton-bisection method). Our system is made up of a unique source that, however, generates different flows since we have only a complete transmission, while the others are patching-derived. In applying the previous algorithm, therefore, every patched flow must be described as generated by a different source. Besides, patching sources usually are high correlated, thus violating a fundamental hypothesis of the Chernoff bound. This algorithm, which is generally conservative [12], [14], may result in an underestimation of the bandwidth requirement for the given loss probability, thus leading to a service failure. Finally, another issue is the lack of knowledge of the distribution of the sources. Usually we use a histogram model [11], [13] to obtain the *marginal distribution* of a source (i.e. the a_{ij}s), and this histogram is stored together with the source flow (and with the schedule derived from the smoothing application). However, in our setting, only the MRCT corresponds to a previously determined histogram, while for the other sources the marginal distributions must be real time computed, since we do not know their schedule until the application of the patching, which begins with a new call.

Now, we present a new CAC scheme that reduces the computational cost compared to the existing one, and that is slightly more conservative then the Chernoff bound method. Besides, it avoids the computation of patched sources histograms, reducing it only to the computation of transmission-no transmission probabilities.

3.2 CAC Variant

Our variant stems from the following observation: the secondary flows (term with which we indicate the flows patched to the MRCT, the primary flow) are made up of periods of no transmission and periods of transmission (whose schedule is exactly equal to the primary flow, but time shifted). If the primary flow is generated by a stationary source (even if we know that this is an approximation), the marginal distribution of the sources of the secondary flows, limited to the transmission periods, is the same of the MRCT. Therefore, we may model a secondary flow as an on-off source that, when is on, with probability p_{on}, transmits to a rate that is a random variable equal to the primary source one, and, when is off, with probability p_{off}, transmits at a null rate. p_{on} can be easily estimated in real time, simply counting the number of intervals in which the server transmits directly to the client and dividing it for N. Let p_i be the probability of being on of the i-th source (i=2,3…m, where m is the number of sources). To extend this notation to the primary source, we assume p_1=1 (i.e. the MRCT is always on). Let n be the random variable representing the number of flows simultaneously on; then $P(n=k)$ denotes the probability of k sources simultaneously on. By the total probability theorem, we have:

$$P(a>c)=\sum_{i=1}^{m} P(a>c|n=i)P(n=i)\,. \tag{12}$$

In this expression, $P(a>c|n=i)$ may be computed with any algorithm. If we use the Chernoff bound, we have i flows generated by the same source; that is, we have to compute only one moment generating function (the primary source one). However, from (9), (11), and (12), we have

$$\lambda=\sum_{i=1}^{m}\frac{e^{-c\theta_i+\Lambda_i(\theta_i)}}{\theta_i\sqrt{2\pi\Lambda_i''(\theta_i)}}P(n=i)=\sum_{i=1}^{m}\frac{e^{-i\frac{M'(\theta_i)}{M(\theta_i)}\theta_i+i\ln(M(\theta_i))}}{\theta_i\sqrt{2\pi i\frac{M''(\theta_i)M(\theta_i)-M'^2(\theta_i)}{M^2(\theta_i)}}}P(n=i)\,. \tag{13}$$

With this equation, we have to consider the m-1 equations derived from

$$c=i\frac{M'(\theta_i)}{M(\theta_i)} \tag{14}$$

for i=1,2…m. Thus, we have a system of m equations in m variables: if we fix λ, we can compute the θ_is from the (13) and the equations derived from (14):

$$\frac{M'(\theta_1)}{M(\theta_1)}=2\frac{M'(\theta_2)}{M(\theta_2)}=...=m\frac{M'(\theta_m)}{M(\theta_m)}\,. \qquad \textbf{(15)}$$

Then we compute *c* from one of the (14). This procedure, besides being extremely expensive, may not admit solution. Actually, (14) violates the hypothesis J_i/c=const. In the next subsections, we present our solutions to these problems.

The first problem we must solve is the estimation of $P(n=k)$ for $1\leq k\leq m$ (since the primary flow is always on, we have at least one transmitting flow, hence we cannot consider k=0). Within the hypothesis of stationary sources and observing that the probabilities to be on or off are independent, since they depend only on the arrival times, it is easy to see that (see [15] for details):

$$P(n=k)=p_1\cdot\sum_{n_1=2}^{m-k+2}\sum_{n_2=n_1+1}^{m-k+3}\cdots\sum_{n_{k-1}=n_{k-2}+1}^{m}\left(\prod_{i=1}^{k-1}p_{n_i}\cdot\prod_{\substack{q=2\\ q\neq n_1,\ldots,n_{k-1}}}^{j}(1-p_q)\right). \qquad \textbf{(16)}$$

We need to compute this expression for all *k* to determine the probability distribution of *n*. However, even for few clients (i.e.: little *m*) the computational cost of such an expression will be too expensive. We try to determine an iterative way to estimate this probability distribution of *n*. Suppose that we have *k*-1 clients instead of *m* and that we know $P(n_{k-1}=i)$ for *i*=1,2…*k*-1, where n_{k-1} denotes the random variable indicating the number of active flows over *k*-1. With this notation, $n\equiv n_m$. If there is a new call, we need to compute the distribution of n_k. We can find an expression of the distribution of n_k in function of n_{k-1} one; indeed:

$$P(n_k=i)=P(n_{k-1}=i)(1-p_k)+P(n_{k-1}=i-1)p_k \qquad \textbf{(17)}$$

for all *i*.

We can proceed in this way until we compute the distribution of n_m, with initial condition $P(n_1=1)=1$. For the details of the algorithm, which is extremely faster then the direct computation of the (16), we refer to [15], where it is also shown that (16) and (17) are equivalent.

3.3 Semi-empiric CAC Algorithm

Although the computation of $P(n=i)$ is very fast and there is only one moment generating function, we cannot use (13) since it is too complex and requires the evaluation of (16). We wonder whether we can have a fast and correct solution if we assume that $\theta_i=\theta_j\,\forall i\neq j$. In particular, it is possible to show that [15]:

$$\sum_{i=1}^{m}\frac{e^{-\Lambda_i'(\theta_i)\theta_i+\Lambda_i(\theta_i)}}{\theta_i\sqrt{2\pi\Lambda_i''(\theta_i)}}P(n=i)\approx\frac{e^{-m\frac{M'(\theta)}{M(\theta)}\theta}}{\theta\sqrt{2\pi\frac{M''(\theta)M(\theta)-M'^2(\theta)}{M^2(\theta)}}}\sum_{i=1}^{m}\frac{M(\theta)^i}{\sqrt{i}}P(n=i) \qquad \textbf{(18)}$$

where $\theta=\theta_m$.

In the next subsection, we will show how simulations confirm the validity of this approximation. Before this, however, we have to show how operates a CAC scheme based on this approximation, and what are its advantages.

When a new call arrives (let us suppose the number m+1), with fixed loss probability λ and multiplexer bandwidth c, we apply a patching scheme to establish a schedule, estimating p_{m+1}. Then we can compute the distribution of n_{m+1} (the number of simultaneously active flows between m+1) in a very fast way starting from the knowledge of n_m and using (17). Now we can solve the equation in θ derived from the equality of λ with the right member of (18). Finally, we can determine the approximate bandwidth requirement c^* from (14) with $i=m$ and compare it width c as for the existing CAC algorithm. If the new request exceeds the threshold value T, a new complete transmission will start.

The computational cost of this algorithm lies in the summation of the right member of (18). However, since there is only one moment to work out, with its derivatives, the complexity is smaller then the original scheme. Besides, we do not need to compute the histograms for each new patched flow, since we can use the already stored one.

3.4 Performance Analysis

In comparing the CAC algorithms with the simulation, we varied various parameters. In all the simulations, we used a histogram model computed as in [12]; besides, we assume that the patching buffers are the same for all clients (it is indifferent for the CAC algorithms), to make clear how this influences the CAC. The patching algorithm used is the TBR, since for our simulations the elaboration time is not important.

In Fig. 6 we can see the comparison between the existing Chernoff bound based algorithm, our variant, and the simulation, varying the patching buffer. In all simulations we assume that the calls after the first are uniformly distributed in a given time interval[4].

The cases shown in Fig. 6 refer to a condition in which the CAC algorithms tend to fail, that is of correlated streams (arrivals are close in time). We can see that our scheme is equal or more conservative than the original Chernoff bound based one. However, it is faster than the latter. This is a conclusion supported by all other simulations we have performed (see [15]). The variant precision, to our knowledge,

[4] Here, the simulations determine only the required bandwidth for a particular realization of an experiment, since is with the same experiment with which the CAC algorithms must be compared. On the contrary, in [12] are performed different simulations for the same mix of films, to compute an average long term bandwidth, but in this case the arrival time does not influence the CAC results (while in our case a change in arrival times influences the schedules).

seems to be influenced by the patching buffer size (an increase in buffer size implies a reduction of the variant precision) but not by the number of bins used for the histogram (which seems to influence only the behavior compared to the simulation) or by the loss probability.

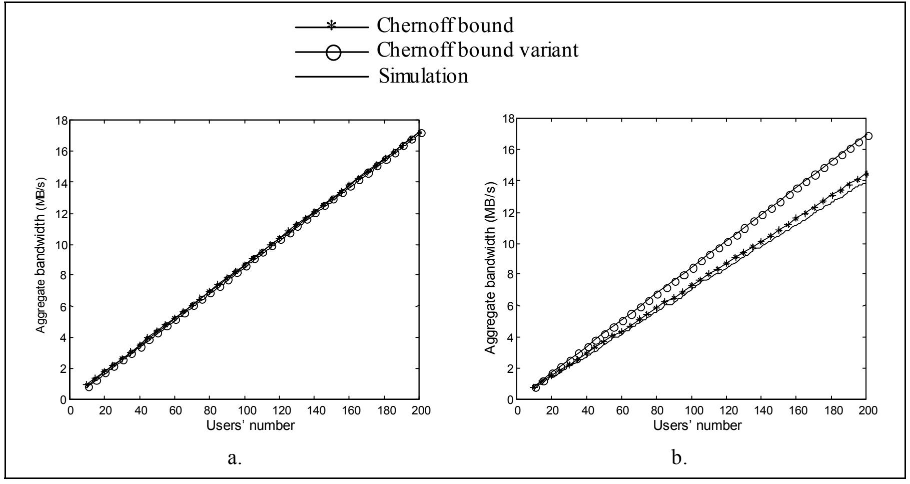


Fig. 6. Aggregate bandwidth estimation for the film "Asterix" with uniformly distributed arrivals within the first 5000 frames, smoothing buffer b=4 MB, 10 bins histograms, $\lambda=10^{-6}$ and different patching buffers: a. B=64 KB; b. B=1 MB. Note how the increased efficiency of the patching in case b. reduces the resulting aggregate bandwidth.

4. Conclusions

The high bandwidth demand, intrinsic to services based on video distribution, makes it attractive to study algorithms for sharing bandwidth resources among the highest possible number of clients. In this paper, we have studied some aspects of these systems in the hypothesis to distribute smoothed stored video, exploiting patching algorithms. Performance indices (efficiency, aggregate bandwidth, etc.) and a statistical Call Admission Control (CAC) have been evaluated by analytical models and validated by discrete event simulation, discussing pros and cons of the various solutions. An extension to multi-source cases of our CAC algorithm has been carried out [15], but, here, it is not reported for lack of space.

References

1. Li, V. O. K., Liao, W., Qiu, X., Wong, E.W.M.: Performance Model of Interactive Video-on-Demand Systems. IEEE JSAC,Vol.14, No.6 (August 1996) 1099-1109

2. Gemmell, D. J., Vin, H. M., Kandhlur, D. D., Venkhat Rangan, P.: Multimedia storage servers: a tutorial and survey. IEEE Computer, Vol. 28, No. 5 (May 1995) 40-49
3. Naussbaumer, J.P., Patel, B.V.: Network requirement for Interactive Video on Demand. IEEE JSAC, Vol. 13, No. 5 (June 1995) 779-787
4. Shachnai, H., Yu, P.S.: On analytic modeling of multimedia batching schemes. Performance Evaluation, Vol. 33 (1998)
5. Lau, S.W., Lui, J.C.S., Golubchik, L.: Merging video streams in a multimedia storage server: complexity and heuristics. Multimedia Systems, Vol. 6 (1998) 29-42
6. Golubchik, L., Lui, J.C.S., Muntz, R.: Reducing I/O demand in Video on Demand storage servers. Proceedings of the ACM SIGMETRICS/PERFORMANCE '95 Conference, Ottawa, Canada (1995)
7. Almeroth, K.C., Ammar, M.H.: The use of Multicast delivery to provide a scalable and interactive Video-on-Demand service. IEEE JSAC, Vol. 14, No. 6 (August 1996) 1110-1122
8. Kamath, M., Towsley, D., Ramamritham, K.: Continuos media sharing in multimedia database systems. Fourth International Conference on Database Systems for Advanced Applications (DASFAA'95), Singapore (1995) 79-86
9. Sen, S., Gao, L., Rexford, J., Towsley, D.: Optimal Patching Schemes for Efficient Multimedia Streaming. Proc. IEEE NOSSDAV'99, Basking Ridge, NJ (June 99)
10. Gao, L., Towsley, D.: Supplying instantaneous Video-on Demand services using controlled multicast. IEEE Multimedia Computing and Systems (June 1999)
11. Salehi, J. D., Zhang, Zhi-li, Kurose, J., Towsley, D.: Supporting stored video: reducing rate variability and end-to-end resource requirements through optimal smoothing. IEEE/ACM Transactions on networking, Vol. 6, No. 4 (August 1998) 397-410
12. Zhang, Zhi-li, Kurose, J., Salehi, J. D., Towsley, D.: Smoothing, statistical multiplexing, and call admission control for stored video. IEEE JSAC, Vol. 15, No. 6 (August 1997) 1148-1166
13. Skelly, P., Schwartz, M., Dixit, S.: A histogram based model for video traffic behavior in an ATM multiplexer. IEEE Transactions on networking, Vol. 1, No. 4 (August 1993) 446-458
14. Knightly, E. W., Shroff, N. B.: Admission control for statistical QoS: theory and practice. IEEE Network (March-April 1999) 19-29
15. Tortorici, M.: Controllo di ammissione in sistemi multicast di flussi video. Dr. Eng. Degree thesis - in Italian (April 2000)

A Migration Path for the Internet: From Best-Effort to a QoS Capable Infrastructure by Means of Localized Admission Control

Giuseppe Bianchi[1] and Nicola Blefari-Melazzi[2]

[1] DIE, Universitá di Palermo, Italy,
bianchi@elet.polimi.it
[2] DIEI, Universitá di Perugia, Italy,
blefari@diei.unipg.it

Abstract. Looking back at many proposals appeared on the scene in these years, a fundamental lesson to be learned is that their success or failure is strictly tied to their backward compatibility with existing infrastructures. In this paper, we consider the problem of providing explicit admission control decisions for QoS aware services. We rely the decision to admit a new flow upon the successful and timely delivery, through the Internet, of probe packets independently generated by the end points. Our solution, called GRIP (Gauge&Gate Realistic Internet Protocol), is fully distributed and scalable, as admission control decisions are taken at the edge network nodes, and no coordination between routers, which are stateless and remain oblivious to individual flows, is required. The performance of GRIP are related to the capability of routers to locally take decisions about the degree of congestion in the network, and suitably block probe packets when congestion conditions are expected. The key message of this paper is that GRIP is a novel reservation paradigm which can be seamlessly applied to the existing Diffserv (and even legacy) Internet, although a marginal increase in QoS is envisioned in these existing scenarios. Indeed, GRIP opens up a future smooth migration path toward gradually improved QoS, as routers in different domain will be upgraded with better measurement-based admission decision criteria. The enabling factor is that router decision criteria are localized and do not involve any coordination. This guarantees that they can be enhanced without losing inter-operability with installed devices.

1 Introduction

The necessity to improve, and possibly to guarantee, the performance perceived by the users is strongly felt in the Internet community. Many proposals have been discussed in the international arena. The reader is probably well aware of RSVP (ReSerVation Protocol). This signaling procedure is the core of the Integrated Services approach [1,2]. It is devised to establish reservations at each router along a new connection path, and provide "hard" QoS guarantees. In this sense, RSVP is far to be a novel reservation paradigm, as it inherits its basic ideas from ATM. RSVP has soft states and it is receiver-initiated while ATM has hard states and it is sender-initiated, but the complexity of the traffic control scheme is comparable. It is indeed true that, in the heart of large-scale networks, such as the global Internet or in high speed backbones of ISPs, the cost of RSVP soft state maintenance and of processing and signaling overhead in the routers is overwhelming. But techniques to reduce the cost of state management, and thus improve scalability have had limited resonance.

Our point is that RSVP, and a number of effective ATM-like solutions as well (think for example to The Label-Switched Path of MPLS, which is not so different from the ATM Virtual Path) may indeed be criticized because of their real or supposed complexity, but probably the lack of their total and ultimate appreciation in the Internet market is simply due to the fact that they are not easily and smoothly compatible with existing infrastructures. What we are trying to say is that complexity and scalability are really important issues, but that backward compatibility and smooth Internet upgrade in an open, un-standardized, market scenario is probably even more important.

Following this line of reasoning, we argue that the impressive success of the Differentiated Services framework [3,4] does not uniquely stays in the fact that is an approach devised to overcome the scalability limits of IntServ. As in the legacy Internet, the DiffServ network is oblivious of individual flows. Each router merely implements a suite of scheduling and buffering mechanisms, in order to provide different loss/delay performance aggregate service assurances to different traffic classes whose packets are accordingly marked with a different value of the DS code-point field in the IP packet header. By leaving untouched the basic Internet principles, DiffServ provides supplementary tools to further move the problem of Internet traffic control up to the definition of suitable pricing / service level agreements (SLAs) between peers[1].

However, DiffServ lacks a standardized admission control scheme, and does not *intrinsically* solve, by any means, the problem of controlling congestion in the Internet. Upon overload in a given service class, all flows in that class suffer a potentially harsh degradation of service. What DiffServ does is to better place the Internet market makers into a position in which pure market choices get reflected in an improved QoS support.

Mapping, to the Internet scenario, the concepts presented in [5] (for a completely different context, i.e. transition from GSM to UMTS), we can envision DiffServ and RSVP as representative of two possible technology migration approaches: (i) an **Evolutionary Approach**, where new services are offered, but the network architecture is not fundamentally changed and over-dimensioning is the solution to improve the performance; and (ii) a **Revolutionary Approach**, where innovative network architectures are developed, at the expense

[1] Which is exactly how the today Internet market operates: compare the free-of-charge ISPs and their eventually unacceptable delay performance, with the expensive "premium" ISPs, where a quite high monthly fare normally guarantees excellent delay performance

of inter-operability with older ones; this correspond to introduce real paradigm shifts.

Both approaches have their pros and cons. In particular, the former is appealing to the market, where the key to quality is left to the strategic initiatives and evolution efforts of each independent provider, but where the lack of advanced architectural solutions renders more difficult to achieve effective performance. The latter is based on an unrealistic hypothesis of having a point in time where a drastic and sudden change should happen, and imposes that all competitors, regardless of their history and previous strategies, need to blindly and fully adhere to an agreed novel standard.

In the middle between these two extremes, we recognize a smoother **Principled Evolutionary Approach**. The key is to recognize a brand new principle, which is revolutionary in its goals and outcomes, but whose ultimate implementation can be delayed in time, by means of subsequent steps of innovation. This approach foresees a continuous evolution in which, in different moments, small and realistic steps of innovation, back-compatible with previous architectures and choices, are asynchronously introduced by different vendors and providers. This requires that a modular and localized concept for innovative features be adopted.

We argue that such a principle, for the Internet, is the use of failed reception of probing packets to discover, at the end points, that a congestion condition occurs in the network, and to reject the new admission request. This idea is extremely close to what TCP congestion control technique does, but it is used in the novel context of admission control. Section 2 describes the distributed components of our proposed admission control mechanism, called GRIP (Gauge & Gate Realistic Internet Protocol), and provides understanding of how the independent GRIP components interact with each other. Section 3 discusses how GRIP is compatible and applicable to the Legacy and DiffServ Internet, and provides preliminary performance figures. Section 4 presents a GRIP implementation in an Internet sub-domain where all routers and traffic sources adhere to a set of hypotheses, and shows that GRIP may ultimately lead to as much as "hard" QoS performance. Conclusive remarks are given in section 5.

2 GRIP: Gauge & Gate Realistic Internet Protocol

The Gauge & Gate Realistic Internet Protocol (GRIP) is a fully distributed and scalable Admission Control scheme, intended to operate over an enhanced DiffServ Internet, but, in principle, compatible with the legacy Internet. GRIP is a constructive answer to a criticism often moved to Diffserv, i.e. that, apparently, DiffServ cannot provide end-to-end guarantees to traffic flows, since it does not provide support (e.g. signaling exchange between routers, and per flow states in routers) to admission control procedures. GRIP builds upon the idea that admission control can be managed by pure end-to-end operation, involving only the new flow ingress router (or source host) and egress router (or destination host). In this, GRIP is related to the family of distributed schemes [6,7,8,9,10,11] recently proposed in the literature under the denomination (following [10]) Endpoint Admission Control (EAC). In addition, GRIP inherits the idea of combining endpoint admission control with measurement based admission control, which was first proposed in [12], where the SRP (Scalable Reservation Protocol) was outlined. Since, at that time, EAC ideas had not yet been published, the authors presented their proposal as a possible solution to the scalability problems of RSVP. Unfortunately (see e.g., what stated in [10]), SRP appeared much more like a lightweight signaling protocol, with explicit reservation messages, rather than an EAC technique with increased intelligence within the end routers. In GRIP, we inherit some key ideas of SRP, but in the light of the brand new paradigm of EAC.

Following a top-down description, we envision GRIP as a mechanism composed of the following three components: (i) GRIP source node protocol, (ii) GRIP destination node protocol, (iii) GRIP Internal Router Decision Criterion. The source and destination node protocols can be considered either running at the ingress and egress nodes (for obvious security reasons) of possibly different ISP. However, from a logical point of view, the source and destination node protocols are more naturally envisioned as running on the user's terminals. We remark that source and destination nodes may belong to different Internet domains, and thus that GRIP must be intended as an Internet-wise admission control solution.

2.1 GRIP Source and Destination Node Operation

The GRIP Source Node Protocol (SNP) is responsible to provide a YES/NO admission control decision upon a new connection setup attempt. The simplest SNP operation is the following. When a user terminal requests a connection with a destination terminal, the SNP starts a *Probing Phase*, by injecting in the network in principle just one *Probe Packet.* Meaningwhile, it activates a probing phase timeout, lasting for a reasonably low time (e.g. from few tens of ms up to few hundreds ms). If no response is received from the destination node before the timeout expiration, the SNP enforces rejection of the connection setup attempt. Otherwise, if a *Feedback packet* is received, the connection is accepted, the probing phase is terminated, and control is given back to the user application which starts a *Data Phase*, simply consisting in the transmission of information packets.

The simplest GRIP Destination Node Protocol (DNP) operation trivially consists in monitoring the incoming packets, intercepting probe packets, reading their source address, and, for each incoming probe packet, just relaying with the transmission of a feedback packet, if the destination is willing to accept the set-up request.

In the basic operation described above, single probe and feedback packets are envisioned, although redundancy (i.e. transmission of more probes and feedbacks) can be considered. In addition, we underline that the described operation does not require at all any source and destination agreement (e.g. standardized signaling information in the probing/feedback packet payloads) on the packet

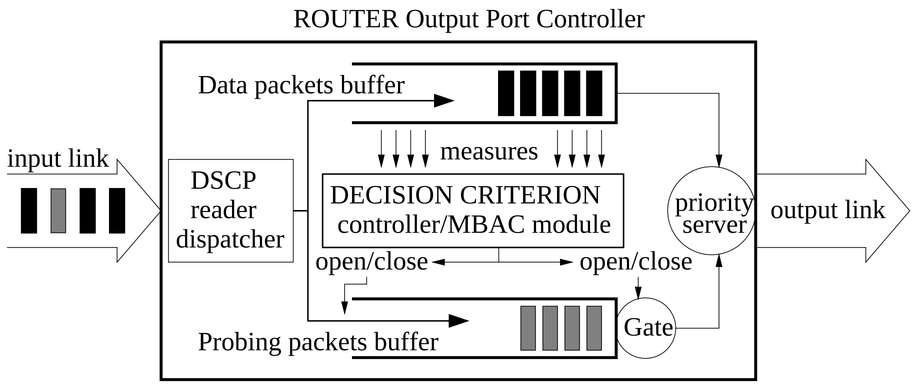


Fig. 1. *GRIP ideal router architecture*

contents, but the whole operation is purely driven by the reception / missing reception of probes and feedbacks.

2.2 GRIP Internal Router Decision Criterion

As mandatory requirement, we impose that probing and data packets can be distinguished ”on the fly” by internal network routers, e.g. that they are tagged with different DS codepoint values in the IP packet header. The ”ideal” GRIP router operation is depicted in Figure 1. For convenience of presentation, we assume that the router handles only the GRIP controlled traffic. Other traffic classes (e.g. best-effort traffic) can be trivially handled by means of additional, separate queues, eventually with lower priority.

At each router output port, GRIP implements two distinct queues, one for data packets, i.e. belonging to flows that have already passed an admission control test, and one for probing traffic. Packets are dispatched to the respective buffers according to the probe/data DSCP tag. The router applies a forwarding priority: probing packets are transmitted only when no data packets are waiting in the buffer. This priority discipline is implemented in most commercial routers and has been indeed specified in the Explicit Forwarding (EF) PHB proposed in DiffServ. It ensures that the performance of the accepted traffic is not affected by congestion occurring in the probing buffer.

Each ideal GRIP router implements an MBAC (Measurement Based Admission Control) module, which measures the **aggregate** data traffic that it is handling. On the basis of the running traffic measurements, the MBAC module implements a Decision Criterion (DC), which continuously drives the router to switch between two states: ACCEPT and REJECT. Several MBAC algorithms have been proposed in the literature (see e.g. [12,13,14] and references therein contained) with the capability to determine, based on running measurements only, whether, at a certain period of time, the considered link is able to accept new connections without QoS degradation, or not.

On the basis of its state, the DC controls the probing buffer server. In particular, when it is in the ACCEPT state, the Probing queue accommodates Probe packets, and serves them according to the described priority mechanism. Instead, when the DC switches to the REJECT state, the router discards all the Probing packets contained in the Probing queue, and blocks all new Probing packets arriving. In other words, the router acts as a gate for the probing flow, where the gate is opened or closed on the basis of the MBAC traffic estimates (hence the Gauge & Gate in the acronym GRIP).

2.3 GRIP Rationale

Each router is locally in charge of deciding whether it can admit new flows, or it is congested. The notion of internal router congestion is not standardized, and it is up to each specific router DC implementation to determine if, and when, congestion arises. The internal (arbitrarily sophisticated and performing) router decision is summarized in the router state (ACCEPT vs REJECT). This state is not notified to the end points by means of explicit signaling information transmission. Instead, end points rely on probing packet losses (i.e. dropped by routers in the REJECT state) as an *implicit* signaling pipe, of which the network remains unaware. In this, GRIP admission control criterion closely relates to the congestion control mechanism of the well known TCP transport protocol, which relies on packet loss information to adapt the source transmission rate.

More into details, when the router is in the ACCEPT state, it advertises that it can admit new connections. This information is implicitly conveyed to the endpoints by allowing probing packets to be served, and thus by leaving them traveling further toward their respective destination. Conversely, when the router is in the REJECT state, no probing packets are forwarded. Since the distributed admission control decision is related to the successful reception at the destination of the Probing packets, and to the consequent relay of feedback packets, locally blocking probing packets implies aborting all concurrent setup attempts of connections whose path crosses the considered router. Conversely, a connection is successfully setup when all the routers crossed by a probing packet are found in the ACCEPT state.

We remark that **no explicit agreement among entities** (e.g. probing packet format, probing/feedback packets payload contents) is necessary to run GRIP. This driving principle is the way to provide a smooth migration path consisting in distributed admission control schemes of increasing complexity and effectiveness, which can indeed operate over a multi-provider and multi-vendor Internet. Each GRIP component accounts for an extremely broad class of possible implementations, and independent implementations of different components can inter-operate. In essence, GRIP guarantees respect for the following principles.

– *Backward compatibility*: as shown in section 3, GRIP may be operated over Legacy or "standard" DiffServ routers.

– *Smooth migration path*: GRIP implementation may start over the actual best-effort Internet to provide marginal performance improvements. A future QoS capable global infrastructure can be provided by independent upgrades.
– *Scalability*: no state information is stored in any router, who handle traffic aggregates and not single flows.
– *Distributed operation*: procedures have a local scope, and each network entity does not have to explicitly co-operate with other entities so that: i) all the network devices can operate autonomously, facilitating multi-vendor markets, ii) the exchange of signaling messages is implicit, iii) the inter-working among different sub-networks/operators is greatly simplified.
– *Performance calibration*: QoS is not granted by the GRIP operation, but will result as a tradeoff between performance and degree of complexity and coordination of the GRIP components. Moreover, there will exist a suitable set of "tuning knobs", i.e. parameters in each GRIP component that allows independent network operators to vary the level of achieved performance and utilization within their domains, and thus set target performance levels.

We now discuss the specific GRIP operation over the legacy and DiffServ Internet, and we follow in section 4 with the description of a full QoS capable Internet domain based on GRIP. These two scenarios are possible extremes of a GRIP-driven migration path that smoothly adds new functionality, thus improving the perceived QoS.

3 GRIP over Legacy and DiffServ Internet

Somehow, we can even say that GRIP is already operating over the legacy Internet. Consider in fact the H.323 call setup scheme based on UDP. In such a scheme, an H.225.0v2 call setup PDU is encapsulated into an UDP packet and injected in the Internet with no guarantee of delivery. In the same time, a state machine is started at the source node. The source node operation is based on a timeout expiration, prior to which a corresponding UDP packet must be received from the destination node. In the protocol, it is possible to set the number of retransmission attempts, which need to be tried before aborting the connection setup, as well as the timeout duration. In other words, the H.323 setup scheme based on UDP simply generalizes, to some extent, the GRIP SNP operation described in the previous section.

Now, GRIP assumes that an MBAC-driven Decision Criterion is implemented within each router. Clearly, legacy routers do not implement any mean to block packets, rather than queue overflow. However, during congestion conditions, all packets (thus including the UDP packet above) suffer of large queueing delay. When the contribute of the queueing delay within the network becomes large, the source node timeout expires before any feedback is received and thus the connection setup is aborted. In essence (as trivially recognized, in the light of the TCP operation) even legacy routers are capable of reflecting internal congestion to the endpoints, as packet queueing delay becomes equivalent to packet loss when the source node timeout operation is accounted for.

GRIP operation can be enhanced over EF DiffServ routers, configured to serve information packets with higher priority than probing packets. Clearly, also in the DiffServ case, routers do not provide any decision criterion, and the router simply delays probing packets upon probing buffer congestion. However, the EF operation has some additional advantages. In fact, the delay experienced by Probing packets is necessarily worse (and thus is a conservative measure) than that experienced by data packets (i.e. belonging to accepted connections). Thus, probes may detect internal router congestion earlier than data packets, and earlier drive reject decisions at the end points. Moreover, the probing buffer congestion is a direct consequence of an increased data traffic throughput. In fact, the EF forwarding discipline operates as a dynamic throttle of the service capacity granted to the probing packets: the greater the accepted traffic, the lower the link bandwidth given to the probing packets, which then experience higher delay and thus imply aborting the relevant setup attempts. This appears to provide a stability feedback: the greater the number of accepted connections, the lower the probability of acceptance for novel connections, and conversely.

Although a full performance evaluation of GRIP over DiffServ is out of the scopes of the present paper, it is indeed instructive to discuss the sample simulation run presented in figure 2. Voice calls, of average duration equal to 1 minute and exponential distribution, are offered to a single link of capacity 2 Mbit/s. The only delay accounted for, in the simulation, is the queueing delay within the router: no propagation delay has been simulated, and the additional delay due to the feedback packet transmission has been set to zero. Voice sources emit according to an exponential ON/OFF pattern, i.e. they alternate exponentially distributed ON periods of average value 1 second, with exponentially distributed OFF periods of average 1.35 seconds. During the ON period, the source emits fixed-sized packets of 1000 bits at a "peak" emission rate of 32 Kbit/s. Trivial computation yields to an average source rate equal to $32 \cdot 0.4255 \approx 13.6$ Kbit/s.

Figure 2 reports the number of admitted voice flows versus the simulation time, for two different load conditions: 110% offered load (i.e. 2.7 calls/s) and 400% offered load (i.e. 9.8 calls/s). Two SNP timeouts have been adopted: an extremely short and unrealistic 10 ms timeout, and a more reasonable 200 ms timeout.

A first consideration is the capability of GRIP to control link overload. With reference to the 400% load case, figure 2 shows that the number of admitted connections frequently overflows the limit value 146.88 (i.e. 100% link utilization). However, as the accepted traffic gets greater than the link capacity, data packets queue builds up, and, thanks to the EF forwarding discipline, the probing buffer server remains blocked until congestion disappears (the "remedy" period, following [13] where a similar behavior is shown to be a characteristic also of centralized MBAC schemes). This proves that, at least, GRIP introduces in plain DiffServ networks an effective and stable form of traffic control, that impedes persistent link congestion. While, in the general case, we are far from satisfactory throughput/delay performance provisioning, table 1 shows that, in the case of light overload such as 110%, better than best effort performance are achieved. In

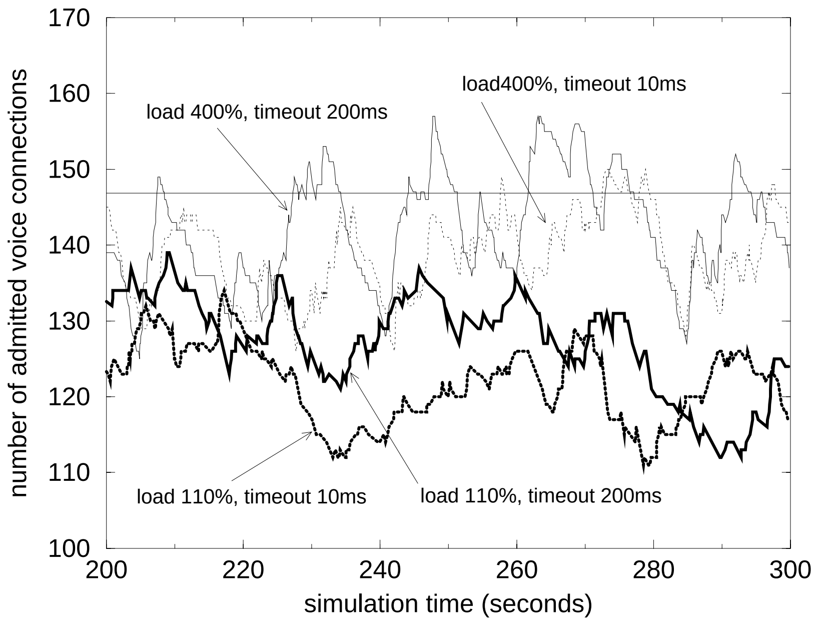


Fig. 2. *Simulation run for GRIP over DiffServ*

other words, GRIP over DiffServ appears to provide a sort of IntServ Controlled Load QoS support.

A second important consideration that can be drawn from figure 2 and table 1 is the role of the SNP Timeout. Intuitively, performance calibration seems possible via SNP timeout tuning, since a short timeout is expected to increase the probability that the setup is aborted before the reception of the feedback packet. Conversely, results show that very strict timeout settings have limited effect on the system performance. Our 10 ms timeout choice was indeed motivated by the attempt to understand whether GRIP over DiffServ could be ultimately tuned to provide toll-quality delay performance (say few ms 99-th delay percentiles, in turns achieved, with the above source and link parameters, by limiting the

Load, Timeout	throughput	95-th delay perc.	99-th delay perc.
110%, 10 ms	0.846 ± 0.002	8.7 ms ± 1.9	51.8 ms ± 12.1
110%, 200 ms	0.902 ± 0.006	61.3 ms ± 7.9	141.3 ms ± 10.4
400%, 10 ms	0.9626 ± 0.0010	229 ms ± 22	402 ms ± 46
400%, 200 ms	0.9806 ± 0.0004	368 ms ± 27	609 ms ± 71

Table 1. *Performance results (with 95% confidence intervals) for the four cases considered in figure 2*

link utilization to about 75% [9], i.e. 110 admitted flows). Unfortunately, in high offered load conditions, the link utilization gets close to 100% (see table 1) despite the 10 ms timeout. Although limited, the spare link capacity left by the EF forwarding discipline to the probing packets appears sufficient to allow occasional periods of very limited queueing delay for probing packets (this particularly happens at the end of a "remedy" period). Hence, probes can thus frequently reach the destination in less than 10 ms and drive flow admissions.

4 GRIP in a Full-Fledged QoS Domain

The major conclusion that can be drawn from the above preliminary investigation is that GRIP over DiffServ is only a preliminary step toward improved QoS support. Ultimately, it appears necessary to upgrade routers with effective decision criterions able to explicitly enforce blocking of probes when the accepted load is in a critical range. Scope if this section is to prove that GRIP can be made able not only to provide *improved* QoS support, but, under specific assumptions, can provide as much as *hard* QoS guarantees within a specific domain[2]. Two key points allow QoS guarantees: traffic control assumptions at the domain edge, and suitable definition of a Decision Criterion to accept or reject probing packets in the routers, based on aggregate traffic measurements.

4.1 Edge Traffic Control

Traffic sources are regulated at the edge of the domain by standard Dual Leaky Buckets (DLB), as in the IntServ framework. The regulated traffic is characterized by three[3] Traffic Descriptors:

- P_S: the Peak Rate (in bytes/s);
- r_S: the Average Rate (in bytes/s);
- B_{TS}: the Token Bucket Size (in bytes).

The DLB guarantees that the traffic offered by a source does not overflow the above specifications. In additions, we specifically require the DLB to enforce that traffic does not underflow the average rate specification. This is accomplished by the emission of "dummy" packets, when the traffic source does not use all the emission opportunities it has. The consequence is that the number of bytes $b(T)$ emitted by a source during an arbitrary time window T (seconds) is upper and lower bounded by:

$$\max(r_S T - B_{TS}, 0) \le b(T) \le r_S T + B_{TS} \tag{1}$$

[2] Such domain is the one proposed in the framework of the project SUITED, sponsored by the European Union in the IST program, and is studied into greater details in a companion paper [17].

[3] DLBs are in more generality characterized by a fourth parameter, called Peak Rate tolerance. For simplicity, it is often assumes, as in [15], that the Tolerance of the Peak Rate is equal to zero or included in the Peak Rate parameter

This property will be extremely important in the definition of a decision criterion able to provide QoS guarantees, i.e. to really achieve the performance promised to the users.

The sources are divided in traffic classes, each comprising independent and homogeneous sources (i.e. with the same DLB parameters). In this paper, we focus on a homogeneous traffic scenario: all the sources have the same DLB parameters. Note that a possible way to handle also heterogeneous sources is the following. DiffServ envisions different "traffic classes", each with specific requirements. In the view of a small number of traffic classes as premium and QoS aware services (e.g. a class could be IP telephony), homogeneous flows with specific requirements make up a class. Each class can be handled in a differentiated way, with its own pair of DS codepoints for probing and data, by means of suitable scheduling mechanisms, similar to those already defined (e.g., WFQ, separate queues).

4.2 Decision Criterion

The localized decision criterion running on each router's output link is based on the runtime estimation of the number of the active offered sources, and on the off-line computation of the maximum number K of sources that can be accepted without exceeding target performance (e.g. loss / delay) levels. For DLB regulated sources, a simple and elegant solution for such an off-line computation has been proposed in [15,16].

For our scopes, we consider K as a "tuning knob", which allow the domain operator to set target performance levels [14]. The operator chooses target levels; the latter are mapped in a value of K and GRIP enforces such value. Thus GRIP is completely independent by the algorithm chosen to evaluate K.

Let's now describe into details the decision criterion adopted. Consider a sufficiently long[4] sliding window T. Each router keeps track of the (time variable) number $A(T)$ of bytes emitted by all the flows that are using the link within the window time.

Assume now that a fixed (unknown) number N of flows is allocated on the link, i.e. no flows arrivals and departures occur. Given a window T and a number $A(T)$ of bytes measured within the window, owing to the bounds (1), N is a random variable in the range:

$$N_{\min} = \left\lceil \frac{A(T)}{r_S T + B_{TS}} \right\rceil \quad \leq \quad N \quad \leq \quad N_{\max} = \left\lfloor \frac{A(T)}{r_S T - B_{TS}} \right\rfloor \tag{2}$$

[4] Specifically, $T \geq T_{\min}$, where

$$T_{\min} = T_{ON} + T_{OFF} = \frac{B_{TS}}{P_S - r_S} \cdot \frac{P_S}{r_S}$$

is the period of the worst case DLB output [15], characterized by an activity (On) period with emission of packets at the Peak rate and a silent (Off) period, both with deterministic lengths.

If no conjecture is made on the statistical properties of the emission process for each source (which depends on how the traffic source fills the DLB regulator, and we do not rely on any hypothesis on the behavior of the sources), the distribution of N between these two extremes remains unknown. Therefore, to provide a conservative estimate, the admission control scheme estimates the number of allocated flows as $N_{\text{est}} = N_{\text{max}}$, and a new flow is accepted if $N_{\text{est}} < K$. In other words the router is in the ACCEPT state and probing packets are allowed to pass the gate, as long as

$$\left\lfloor \frac{A(T)}{r_S T - B_{TS}} \right\rfloor < K \tag{3}$$

Clearly, the longer T, the narrower the range in unequality (2), and the higher is the link utilization. In fact, whenever $N \geq N_{min}$, condition (3) may result unsatisfied, thus leading the router to switch to the REJECT state even if $N < K$ and a new flow might be accommodated. However, a long value T has an important drawback. In fact, the above analysis has been concerned with the idealistic case of a constant number of admitted flows. When a realistic dynamic flow allocation occurs, a short measurement window is needed in order to react quickly to the variation of the number of active flows due to the flows arrivals and departures. Quantitative considerations related to the optimal dimensioning of T are carried out in a companion paper [17].

4.3 Transient Management and Stack Protection

An extremely important problem, dealt with into details in [17], is the management of transient and potentially critical situations occurring when new flows activate. Consider, in fact, a router in the ACCEPT state. Probing packets crossing the router, and arriving at the destination, result in new flows activation. The extra load of an activating flow is not fully accounted by the pure measurement $A(T)$ and the decision rule (3), until the newly activated flow has transmitted for at least T seconds. This implies that, when a large number of new flows activate in a very short time frame, overallocation above the maximum value K may occur, and thus QoS is not guaranteed in all operational conditions.

To account for this problem, we introduce a protection scheme based on a "stack" variable, which keeps memory of the amount of "transient" flows. Whenever a probe packet is transmitted, the stack is incremented by one. A timer equal to the duration of the measurement window T is then started, and the stack is linearly decremented at a rate $1/T$ until the timer expires. The value of the stack changes the router gate condition (3) as follows: the router is in the ACCEPT state and probing packets are allowed to pass the gate, as long as

$$\left\lfloor \frac{A(T)}{r_S T - B_{TS}} + \text{STACK} \right\rfloor < K \tag{4}$$

4.4 A Test Case Simulation Run

We consider traffic sources controlled by a DLB with parameters[5]:

- $P_s = 4000$ bytes/s;
- $r_S = 1700$ bytes/s;
- $B_T S = 5300$ bytes;

offered to a router link with rate $C = 250000$ bytes/s and buffer size $B = 53000$ bytes. By setting a target loss probability equal to 10^{-5}, it results, according to [15] that the maximum number of such DLB-shaped sources that can be admitted to the link is $K = 100$, resulting in a maximum link utilization equal to 68%.

The ultimate target of the described GRIP operation is to achieve a link utilization as close as possible to the above maximum, but under the very strict QoS requirement of **never** exceeding it, i.e. never admitting a number of sources greater than $K = 100$. In order to satisfy the latter (quite restrictive) requirement, we expect a throughput penalization, substantially due to the following reasons:

- *conservative estimation of the number of accepted flows.* For a window time $T = 20$, used in the simulation runs, and the above DLB parameters, it results that, when $A(T)$ is such that $N_{\max} = 100$, $N_{\min} \approx 73$. This means that when the number of admitted flows exceeds this value, there is a non null probability (depending in a non trivial manner on the specific source emission pattern) that the router is found in the REJECT state.
- *Stack protection and lost probing packets.* The drawback of the stack technique is that subsequent routers along the path may discard some of the probing packets. In this case, the stack will provide transient reservation of system resources for a non-existent flow. Although such "preventive" stack reservation linearly decays with T, we expect some impairment on the system performance.

We have considered a scenario where a router is loaded with new calls arriving at (Poisson) rate 3 calls/s. Each call, when accepted, transmits at constant rate[6] r_S, and has duration drawn from an exponential distribution with mean value 4 minutes. This implies that 720 Erlangs are offered to the link, i.e. more than 7 times the maximum number of calls that can be, in principle, simultaneously admitted (K=100).

In addition, to stress the GRIP operation and the stack protection mechanism, we have assumed that probing packets, once served at the router, may be discarded in the remaining network path. In particular, in the simulation run,

[5] The choice of such DLB parameters have here only a significance of a case study. Such case study is targeted to the SUITED system parameters [16].

[6] Results regarding exponential on/off calls, presented in [17], show that the resulting system utilization is higher in the on/off case. Refer also to this paper for quantitative considerations on the selection of the value T.

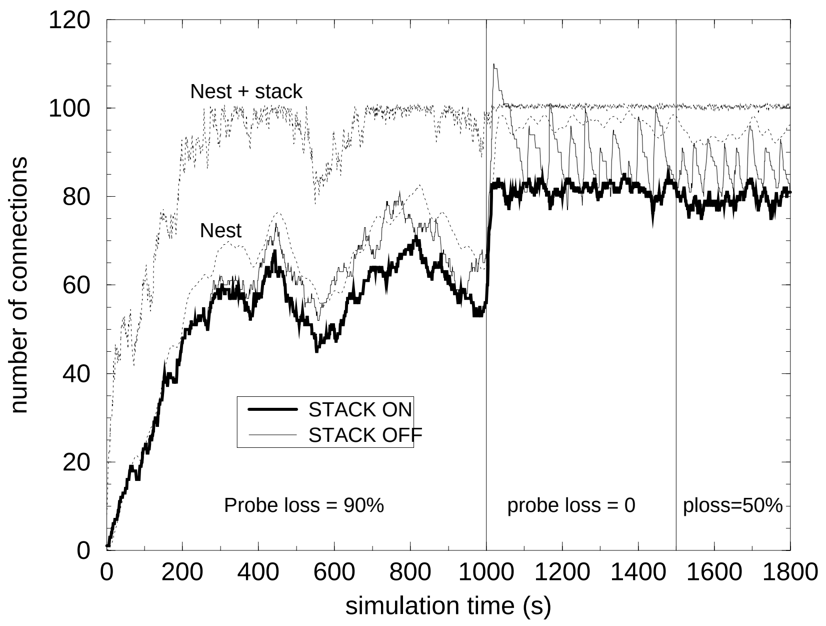


Fig. 3. *full-fledged GRIP simulation run*

we have set in the first 1000 seconds a probe discarding probability as high as 90%, meaning that, on average, only one probing packet out of 10 transmitted by the router will result in an active connection. At 1000 seconds, we have suddenly switched to a 0% probe loss, thus abruptly loading the router with active calls. Finally, at 1500 seconds, we have introduced a probing packet discarding probability equal to 50%.

The results of the described simulation scenario are reported in figure 3. The figure reports the number of admitted connections versus the simulation time, for both cases of stack protection mechanism active and non active. In addition, for the case of stack on, the runtime router conservative estimation (3) of the number of admitted connections is reported under the label "Nest", as well as this estimation including the stack ("Nest+stack"), as adopted in (4).

The analysis of the figure allow several important considerations. First, we see that, in the first 1000 seconds of simulation, the stack protection mechanism slightly penalizes the system throughput. While some calls are blocked when the stack operates (this happens when Nest+stack reaches 100), no calls blocking has been observed with the stack mechanism off (note that the load in terms of active calls is only 72 erl, as 90% of the probes are discarded). However, as shown by the simulation at time 1000 seconds, the stack mechanism is indeed necessary to avoid overload. In fact, without stack protection, we see that the number of accepted calls overflows the maximum value 100, and activates a load oscillation in which the system shows periodic peaks of high load (refer to [14] for a thorough understanding of such behavior in MBAC systems in general).

A second consideration is the efficiency of GRIP. Such efficiency lies in minimizing the difference between the admitted number of sources and the correspondent theoretical maximum value K. After 1000 seconds, we see that the number of admitted connections stabilizes around 80, which is about 20% less that the maximum. This performance degradation is mostly due to the conservative estimation of the number of admitted connections, which is shown in the figure to stay around 95. Clearly, we remark that all design choices were driven by the necessity of an extremely robust scheme providing strict performance guarantees. However, we stress that the performance of GRIP must be evaluated in its ability to enforce a specific value of K. The throughput efficiency is a direct consequence of the selected value of the "tunable knob" K, and can be thus arbitrarily adjusted by the network operator, who can increase the system utilization at the expense of strict QoS guarantees.

It is very interesting, and quite surprising, to note that, after time 1500, the introduction of a 50% probing discard ratio only marginally affects the throughput performance. This shows that, with the exception of very critical conditions (e.g. the 90% probe loss considered in the first part of the simulation), the increased inefficiency due to the stack mechanism is very limited.

5 Conclusions

This paper has presented GRIP (Gauge&Gate Realistic Internet Protocol). The most important message we have tried to convey is that GRIP is not a new reservation protocol for the Internet (in this, differing from the SRP protocol [12], from which GRIP inherits some strategic ideas). Instead, GRIP is a novel reservation paradigm that allows independent end point software developers and core router producers to inter-operate without explicit protocol agreements.

The principle at the basis of the GRIP operation is to enforce admission control decisions to operate on the basis of the reception of (or lack of) probing packets, injected in the network before a connection setup. In doing this, GRIP, in a certain sense, extends the principle at the basis of TCP to the completely different and novel problem of providing explicit per-flow admission control over stateless Internet architectures.

We have shown that GRIP can be adopted over legacy and DiffServ Internet, although, as shown in section 3, without QoS guarantees. However, in our view, this is not a problem, as GRIP effectiveness can be enhanced with independent core router Decision Criterion upgrades. In this sense, GRIP is a very appealing paradigm in terms of mass-market development, since all the required modifications in the migration path toward toll-quality QoS support are incremental, and may be independently offered by different vendors.

Finally, as an example, of its ultimate capabilities and of the effectiveness of purely localized operation, we have presented a particular GRIP implementation which, on the basis of suitable traffic assumptions and router operation (to be considered applicable within a specific IP domain), allows as much as hard performance guarantees.

References

1. R. Braden, L Zhang, S. Berson, S. Herzog, S. Jamin, "Resource ReSerVation Protocol (RSVP) - Version 1 Functional Specification", RFC2205, September 1997.
2. J. Wroclawsky, "The use of RSVP with IETF Integrated Services", RFC2210, September 1997.
3. K. Nichols, S. Blake, F. Baker, D. Black, "Definitions of the Differentiated Service Field (DS Field) in the Ipv4 and Ipv6 Headers", RFC2474, December 1998.
4. S. Blade, D. Black, M. Carlson, E. Davies, Z. Wang, W. Weiss, "An Architecture for Differentiated Services", RFC2475, December 1998.
5. E. Berruto, G. Colombo and A. Napolitano: "From Gsm to Umts: a Continuous Evolution with Steps of Innovation", 1st European Personal and Mobile Communications Conference (EPMCC 95), Bologna, Italy, 28-30 November 1995.
6. F. Borgonovo, A. Capone, L. Fratta, M. Marchese, C. Petrioli, "PCP: A Bandwidth Guaranteed Transport Service for IP networks", IEEE ICC'99, June 1999.
7. V. Elek, G. Karlsson, "Admission Control Based on End-to-End Measurements", Proc. of IEEE Infocom 2000, Tel Aviv, Israel, March 2000.
8. G. Bianchi, A. Capone, C. Petrioli," Throughput Analysis of End-to-End Measurement Based Admission Control in IP", Proc. of IEEE Infocom 2000, Tel Aviv, Israel, March 2000.
9. G. Bianchi, A. Capone, C. Petrioli: "Packet management techniques for measurement based end-to-end admission control in IP networks", IEEE/KICS Journal of Communication Networks, June 2000.
10. L. Breslau, E. W. Knightly, S. Schenker, I. Stoica, H. Zhang: "Endpoint Admission Control: Architectural Issues and Performance", ACM SIGCOMM 2000, Stockholm, Sweden, August 2000.
11. R. J. Gibbens, F. P. Kelly, "Distributed Connection Acceptance Control for a Connectionless Network", 16th ITC, Edimburgh, June 1999.
12. W.Almesberger, T.Ferrari, J. Y. Le Boudec: "SRP: a Scalable Resource Reservation Protocol for the Internet", IWQoS'98, Napa (California), May 1998.
13. M. Grossglauser, D. N. C. Tse: "A Time-Scale Decomposition Approach to Measurement-Based Admission Control", Proc. of IEEE Infocom 1999, New York, USA, March 1999.
14. L. Breslau, S. Jamin, S. Schenker: "Comments on the performance of measurement-based admission control algorithms", IEEE Infocom 2000, Tel-Aviv, March 2000.
15. A. Elwalid, D. Mitra: "Traffic shaping at a network node: theory, optimum design, admission control", IEEE Infocom 97, pp. 445-455.
16. N. Blefari-Melazzi, M. Femminella, G. Reali: "Dynamic Bandwidth Allocation in a Circuit- Switched Satellite Network: Provision of Deterministic and Statistical QoS guarantees", IEEE Infocom 2000, Tel-Aviv, Israel, March 2000.
17. G. Bianchi, N. Blefari-Melazzi, M. Femminella: "GRIP: QoS support over Stateless DiffServ Networks by means of localized measurements and decisions", submitted for publication.

Author Index