# Performance evaluation and comparison of protocol stacks providing Quality-of-Service in the Internet*

Elena Pagani, Gian Paolo Rossi
Dipartimento di Scienze dell'Informazione
Università degli Studi di Milano
via Comelico, 39. I-20135 Milano, Italy
e-mail: {pagae,rossi}@dsi.unimi.it

**Abstract**

In this paper we present some performance evaluations and comparisons of different protocol stacks proposed in the literature to support Quality-of-Service (QoS) in the Internet. With the aim of performing the comparison, we realized a simulation framework that allows to easily add, substitute and modify the modules implementing the network protocols under test. The framework structure derives from the functional architecture we have previously characterized, for the support of QoS in large multi-domain networks where both integrated services and differentiated services are deployed. The framework is compatible with the standards proposed in the literature. Both unicast and multicast communications are supported.

In this work, we briefly review the reference functional architecture to which the framework is inspired. Then, we describe the framework structure and the functional modules that have been implemented and inserted into it. Finally, we discuss the performance measures obtained using the framework. These evaluations allow to gain a better comprehension of both the contributions given by the different network protocols in the architecture and their relationships. They are demonstrating to be very helpful in the deployment of a testbed environment which will allow to perform more extensive experiments over a real network by modularily combining software objects either available from public domain or implemented ad hoc.

**Keywords:** Quality-of-Service, multicast, multimedia applications, system architecture, performance evaluation.

## 1 Introduction

The tremendous diffusion of the Internet is making IP an attractive platform to support multimedia traffic too. Unfortunately, despite the continuous growth of the link capacity

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

and of the switching power of the routers, the best effort nature of IP is not sufficient, alone, to ensure the required quality to that traffic. Several initiatives have been undertaken in recent years to bridge this *Quality Of Service* (QoS) gap. Protocols like RSVP [31], RTP [24] or more recently QoSMIC [14], are just a few examples of the efforts in this direction, while *integrated services* (int-serv or IS, for short) [10, 30, 26] and *differentiated services* (diff-serv or DS, for short) [9, 7] are the most notable models of service differentiation which have been defined by the IETF Working Groups.

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.

For the purpose of characterizing good standard architectures and protocols, extensive experimentations must be performed to study the behaviours of the solutions proposed in the literature. Those experiments allow to highlight the impact that different approaches have on the QoS performance indexes. By means of experimentations, a better comprehension can be achieved about the mechanisms needed at each architectural level to provide QoS, and how those mechanisms should inter-operate. The experiment results should guide in both $(i)$ characterizing what mechanisms must be used to provide a given QoS level; and $(ii)$ devising new mechanisms when the existing ones are not suitable to obtain the desired behaviours. Unfortunately, appropriate tools to perform those experiments do not exist as yet. The implementation of testbeds over real networks has been planned (e.g., the QBone initiative [19]), but the needs of characterizing the methodologies to adopt to perform the experiments is slowing down the achievement of the intended objectives. On the other hand, the existing simulation tools offer little support to the study of QoS-sensitive protocols.

With the aim of studying in depth the problems involved with the QoS support in the Internet, our group has performed the performance evaluation and the comparison of the behaviours of different protocols and mechanisms proposed in the literature to realize QoS. This task has been carried out by means of simulation techniques: we implemented a simulation framework for the testing and evaluation of protocols for end-to-end QoS in the Internet, on top of the NS-2 [13] simulation package. As the starting point, we devised a reference functional architecture [22]. The architecture is consistent with the standards proposed in the literature, and can support both unicast and multicast communications. Moreover, it is compatible with the existing QoS-sensitive protocols. This architecture has been taken as the reference in the deployment of our simulation framework. The framework provides a common experimentation environment to which the QoS initiatives can refer to cooperate for the construction and testing of a QoS architecture. It supplies the building blocks to assemble to build such an architecture. For each block, several policies are implemented that realize its functionalities, thus allowing to perform experiments with different system configurations. We carried out several tests by combining the implemented modules in various ways, and considering different system settings. The achieved results allow to gain a better comprehension of both the contributions given by the different network protocols in the architecture and their relationships. They are demonstrating to be very helpful in the deployment of a testbed environment which will allow to perform more extensive experiments by modularily combining software objects either available from public domain or implemented ad hoc.

The paper is organized as follows: in section 2, we give an overview of the functional architecture [22] that has been taken as the reference in the framework construction. In section 3, we discuss in detail the structure of the proposed simulation framework , and we describe the implemented modules that can be inserted into the framework to perform experiments. In section 4, we report the performance measures we have obtained using the proposed framework. In section 5, some final remarks are discussed.

# 2 QoS functional architecture

In [22] we propose a reference functional architecture to support QoS in the Internet. Such architecture is consistent with the proposals of standards existing in the literature. It has been designed for large networks, where both integrated services (int-serv's or IS, for short) [10, 30, 26] and differentiated services [9, 7] (diff-serv's or DS, for short) can be deployed. It supports both unicast and multicast communications. It is composed by the functional modules needed to provide QoS over an IP-based network. Most part of the functional modules composing the described architecture are covered by the unicast-oriented and multicast-oriented protocols which have been proposed so far [24, 31, 25, 5, 18]. Other modules are available within either research or proprietary QoS architectures (e.g., MBone, XRM, the Tenet architecture, the Heidelberg QoS model) as described in [2]. 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 [29]. Broadly speaking, the functions to guarantee end-to-end QoS are located on top of the transport layer, where the UDP protocol is used, while at level 3 we locate the functions to manage the network resources and the QoS traffic streams over the path from the source to the destination(s). The functions at level 2 implement the packet forwarding policy over the selected links.

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; the greek symbol shown with each module indicates to which class it belongs.

**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 specification into the format which

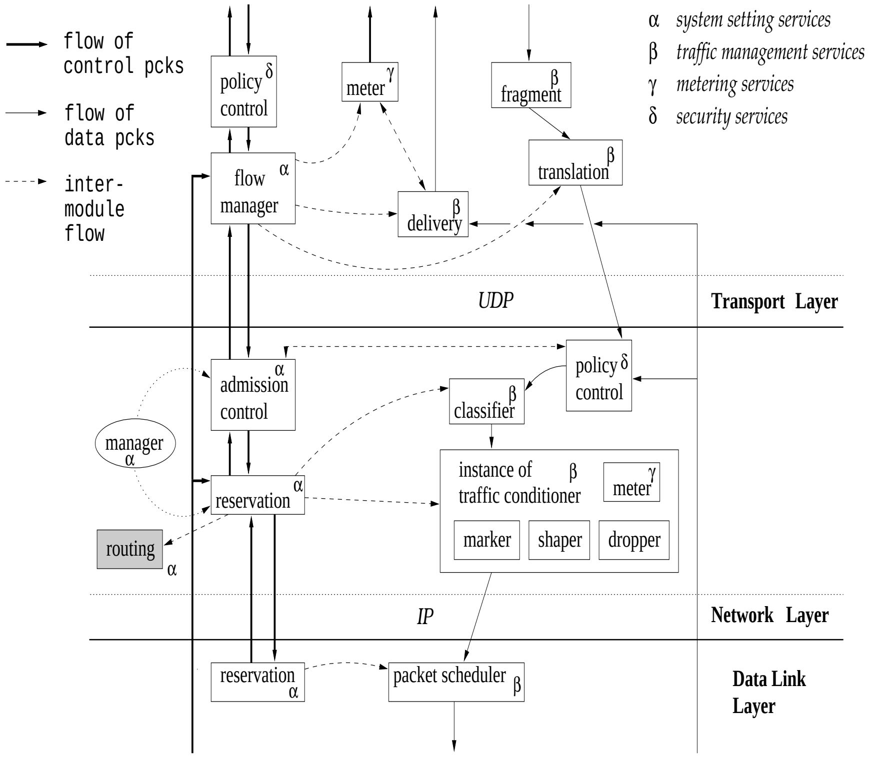



Figure 1: Functional architecture of a QoS-capable router.

is suitable for the network layer (e.g., a `TSpec` and a `RSpec` [27, 28] in the case of int-serv's, or a Per-Hop-Behavior (PHB) 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. At the end-to-end layer, the destination may periodically report upstream to the source peer-entity the observed QoS variables which include, for example, the transmission delay and the message loss rate [24]. The results can be used to dinamically re-configure some of the functional modules for traffic management to adapt to the changing network conditions. The

reports might be as well notified to the source application, to allow traffic profile adaptation according to the available network resources.
Metering is used at the network layer to measure the characteristics of the packet streams which belong to a given traffic class, and to compare them with the traffic profile of that class. The measurements may be used to dynamically adapt the resource reservation and the packet scheduling policies.

**Security class.** This class includes functions which allow the authentication of an application, access control, accounting and security management. QoS will be probably charged more than the best effort service. Attacks such as denial-of-service or theft-of-service should be prevented. Those issues justify the need for security services. At the end-to-end layer, the authorization of an application/user to access a QoS service is controlled. Information may be recorded for accounting purposes.
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, either from an adjacent router or from the upper layer, 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 [22]. In that work we show as well how the proposed architecture can be applied to provide interoperability among different domains, by considering the border router architecture and describing the intra-module communication flows. We study the case of a QoS traffic flowing both through the IS-DS-IS domain configuration and through the DS-IS-DS configuration.

## 2.1 Supporting QoS-multicast traffic

The most part of the multimedia applications are *multicast*, i.e., they are characterized by a *one-to-many* or *many-to-many* communication pattern. And they are QoS-sensitive too. The capability of supporting multicast communications is useful to achieve an efficient use of the network resources [11], however, the market receptiveness for such a service heavily depends on the ISPs ability to provide a degree of quality higher than the best effort nature of current multicast protocols for the Internet (e.g., PIM [12] and CBT [4]). So far, very few proposals have been presented to provide multicast QoS (e.g. RSVP [31], RTP [24], QoSMIC [14]). All these proposals consider the multicast within a single domain. Inter- and intra-domain communications are not considered in details.

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.

Different receivers may be characterized by different resource availability, thus requiring different levels of QoS. As the consequence, to deal with the receivers heterogeneity, packet processing may be performed according to different policies for different destinations. A given routing entity belonging to a multicast tree infrastructure may be either a leaf router or be connected to one or more downstream entities through downstream links (i.e. it is the

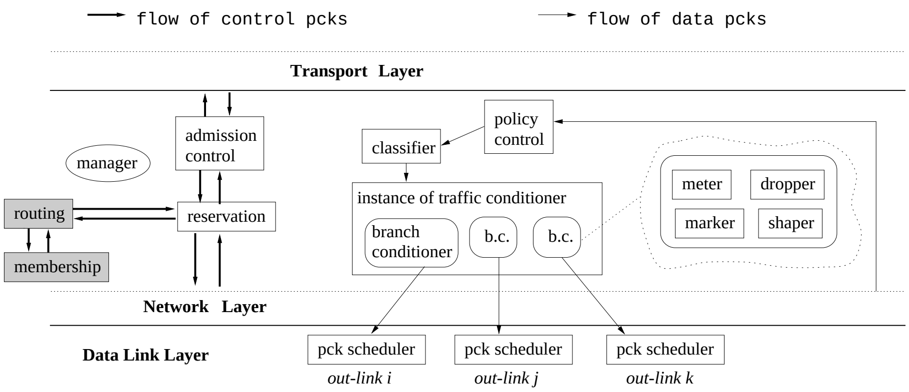



Figure 2: QoS multicast functional architecture.

parent of one or more sub-trees). The quality of path of the downstream links may differ. 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 instance, the intermediate routers may resort to utilize different data encodings/compressions over different tree branches.

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. Metering must be performed by gathering QoS parameters from the destinations. The measurements are notified to the application, which decides how to behave according to application-dependent arguments. If *fair delay* requirements [3] 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* 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 [22].

# 3 Implementation of the simulation framework

Many possible implementations have been devised and proposed in the literature to realize the functional modules composing the described reference architecture. The use of simulations can be helpful to understand how the different functional modules impact on the performance in terms of the QoS parameters. Starting from the NS-2 simulation package [13], 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 [3], 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 [24] is available for what concerns QoS support.

The simulation framework is intended to ease the development and testing of new QoS-sensitive protocols, by supporting a sort of *programming by components*. It currently involves the basic modules for each architectural level; some alternative implementations have been realized for each module. Modules implementing standard protocols have been encapsulated into the framework. The modules can be variously combined to perform experiments under different system settings. Measurement tools have been included.

In Figure 3 we show the architecture under test; on the right side the NS-2 classes are reported, which have been used (possibly modified) or created to realize the simulation framework.

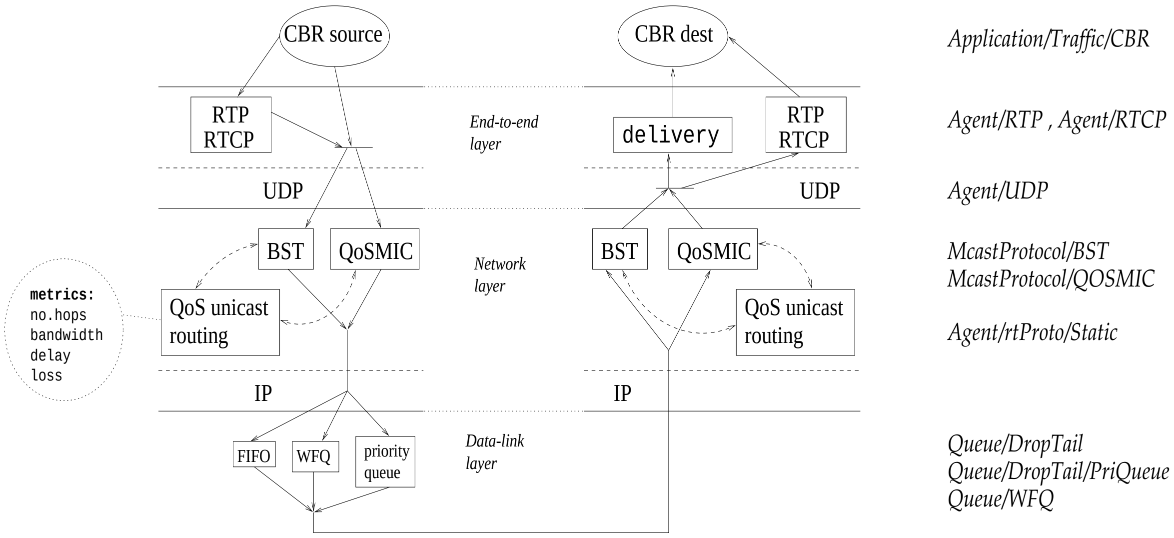



Figure 3: Layout of the implemented architecture.

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;

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

We are extending the framework with other policies either already available in NS-2 (e.g., RED [16], CBQ) or that we are currently implementing, such as WRR [21] and WF2Q [6].

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 [14, 5], which has been chosen among others [18, 8] for its simplicity and flexibility. QoSMIC builds a shared tree $T_G$ composed of the receivers belonging to a multicast group $G$. 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* metric at a time; it requires an underlying QoS-sensitive unicast routing protocol that finds "good" paths according to that metric. The QoSMIC protocol can be used to provide resource reservation in both IS and DS domains. In the latter case, reservation is performed for the aggregates of micro-flows; both static and dynamic system re-configuration is supported.

Besides of QoSMIC, other multicast routing protocols are natively available with NS-2, such as Bidirectional Shared Tree (BST), DVMRP [23], PIM-SM [12] and PIM-DM. In the future developments of our framework, we plan to modify the NS-2 classes for BST and PIM-SM to implement respectively the QoS CBT protocol [18] and the QoS PIM-SM protocol [8].

At the end-to-end layer, we modified the NS-2 RTP class to meet the real implementation of that protocol [1]. We interfaced it with the underlying routing protocols. We are as well designing an end-to-end signalling protocol that can be used either instead of or in combination with QoSMIC. Such a protocol should be run below RTP.

Throughout the framework deployment, we kept in mind the objective of modularity. Hence, the described modules can be freely composed to perform testing, as we have verified in our experiments; the obtained results are discussed in the next section. These evaluations allow to gain a better comprehension of both the contributions given by the different network protocols in the architecture and their relationships. They are demonstrating to be very helpful in the deployment of a testbed environment which will allow to perform more extensive experiments by modularily combining software objects either available from public domain or implemented ad hoc.

# 4 Preliminary simulation results

We are currently running simulations for CBR traffic; the achieved results are reported in this section. The experiments have been performed with all of the packet scheduling policies described in the previous section. We are considering the received *rate* as the unique QoS metric; hence, the quality-of-path is expressed in terms of the available bandwidth. As shown in figure 3, the source of CBR traffic accesses the QoSMIC service through the UDP protocol. QoSMIC is below UDP and on top of a QoS unicast protocol, Distance Vector-based, which is on top of IP. The default `delivery` module delivers the packets as soon as they are received; RTP can be used in place of it.

In our experiments we consider a flat network of 64 routers interconnected by optical fiber links having a 2 Mbps bandwidth and length variable between 50 and 100 Km. QoS traffic originates in the tree root $R$; we performed our measures with different transmission rates. The background, best effort traffic can be either uniformly distributed all over the network or deterministically sent along a tree branch; the average bandwidth occupancy is

33%. As the consequence, the contention between QoS and best effort traffic initiates when $R$ generates traffic at about 1.3 Mbps. The upper bound on the packet queue length is 20.

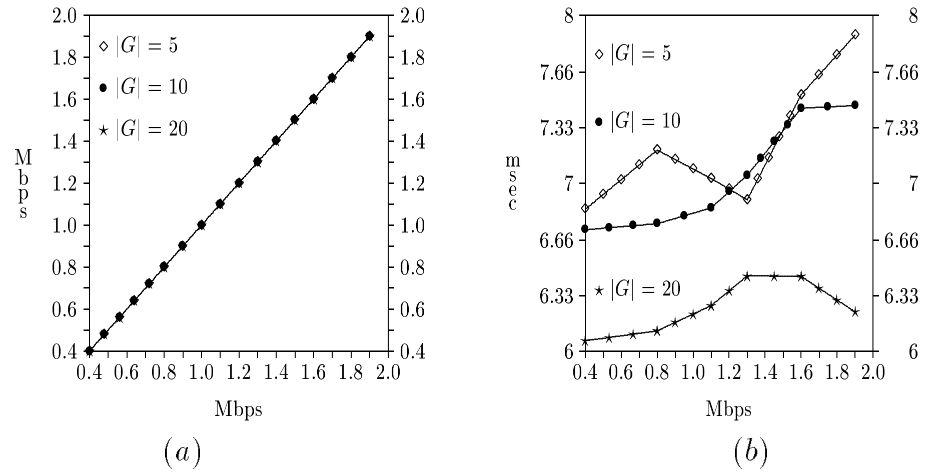



Figure 4: ($a$) Received rate vs. offered QoS load for different group cardinalities. ($b$) Average end-to-end delay.

We evaluated the performance of the protocol stack ⟨QoSMIC, WFQ⟩ for different group cardinalities, in the presence of uniform background best effort traffic. Bandwidth reservation is an effective mechanism to guarantee the delivered rate (Figure 4($a$)), independently of the number of receivers. Hence, both delay (Figure 4($b$)) and fair delay (Figure 5($a$)), that is the difference between the end-to-end delays perceived by the slowest and the fastest destination, are not affected by the background best effort traffic. The reservation, emulated by the WFQ policy, bounds the queuing delay for the QoS packets. The end-to-end delay is then roughly the sum of the transmission and propagation delays. As the consequence, the average delay only depends on the distance from the source: for high group cardinality it decreases because the receivers are more dense in the root neighbourhood. The fair delay as well depends on the system topology: for large groups it increases with the distance between the nearest and the farest destination from the root.

By contrast, background traffic has a negative impact on the jitter (Figure 5($b$)) in spite of the bandwidth reservation. We measured a jitter of the order of $10^{-18}$ msec. for each destination in the absence of best effort traffic. While, when best effort connections overlap tree branches, the jitter measured by the destinations grafted on those branches increases up to $10^{-1}$ msec. For all group cardinalities the jitter behaviour is chaotic. Its decrease for high CBR rates depends on the best effort traffic being excluded from the network, because virtually all of the bandwidth is reserved for the QoS traffic. This case, the interference between the two traffic classes disappears.

When using RTP as the `delivery` module, the jitter decreases to zero. Our current implementation of RTP buffers the QoS packets at the destination, and delivers them at a constant rate after a fixed, a-priori chosen delay greater than the maximum transmission delay. As the consequence, the observed end-to-end delay grows up to that upper bound. To achieve a better efficiency, an estimate of the transmission delay could be obtained from the RTCP reports.

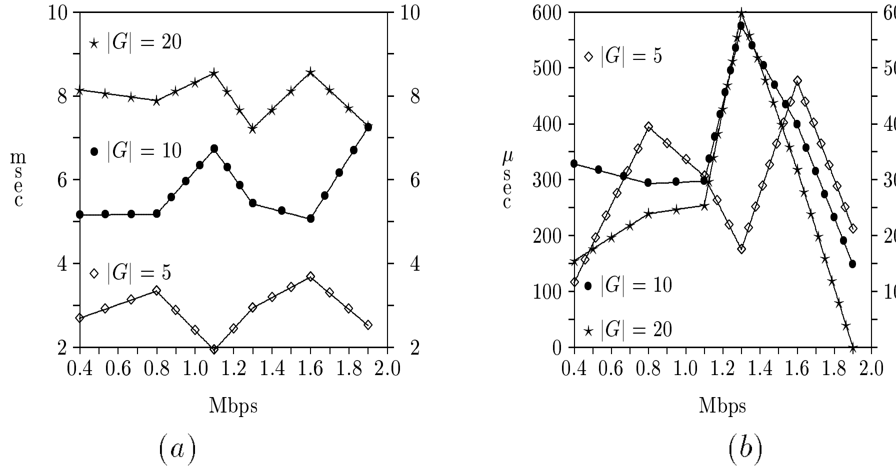



Figure 5: ($a$) Fair delay vs. offered QoS load for different group cardinalities. ($b$) Average jitter.

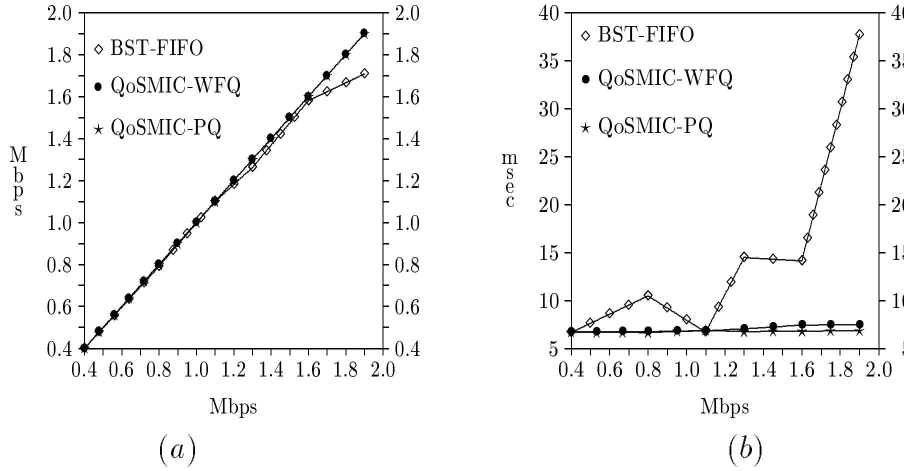
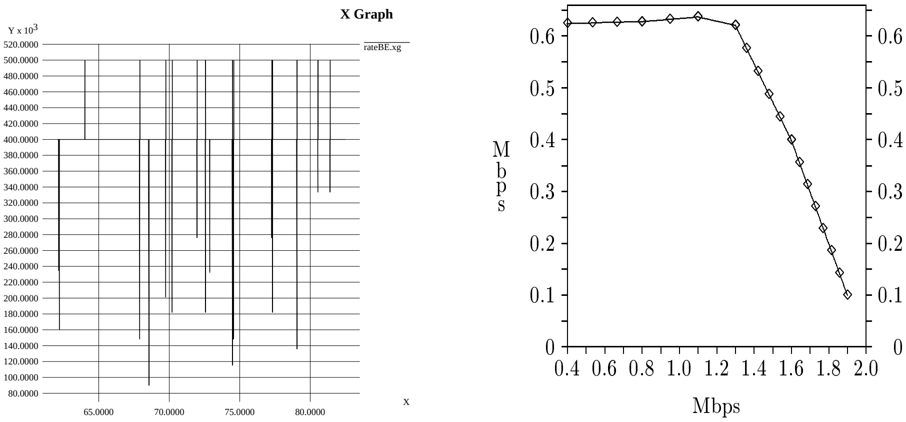



Figure 6: ($a$) Received rate vs. offered QoS load for different packet scheduling policies. ($b$) Average end-to-end delay.

To test the impact of the packet scheduling policy, we compared the performance achieved by QoSMIC when using either WFQ or PQ. As the reference measure we considered that obtained with the stack ⟨BST, FIFO⟩, that is, without performing resource reservation. Also in this case, the background best effort traffic is uniformly distributed all over the network. Only the results obtained with group cardinality equal to 10 are shown. With a regular traffic flow produced by the QoS source, WFQ and PQ achieve comparable results. By contrast, if no bandwidth is reserved, the contention with the best effort traffic highly penalizes the QoS traffic in terms of both the delivered rate (Figure 6($a$)) and the average end-to-end delay (Figure 6($b$)). QoS packets are queued and they can be dropped as the consequence of buffer overflow. The fair delay figure is similar to that of the delay, while the jitter is chaotic in all cases unless RTP is used.

We observed the impact of the resource reservation on the best effort traffic under the same conditions, using the stack ⟨QoSMIC, WFQ⟩. While the number of queued QoS packets is bounded above by 1, the best effort packets soon saturate their portion of the buffer and are then dropped. In Figure 7($a$) we show the best effort throughput (in bps) at a node that




($a$) ($b$)

Figure 7: ($a$) Best effort rate (bps) vs. time (sec.). ($b$) Average best effort throughput vs. offered QoS load.

is destination of both the QoS flow and a best effort connection, over 20 sec. of simulation time, when the CBR source generates traffic at 1.6 Mbps. The best effort traffic uses the residual bandwidth that is not reserved for QoS packets, thus achieving a mean rate around 0.4 Mbps. This can be as well observed in Figure 7($b$), where the best effort rate is shown with respect to the offered QoS load. The figures 7($a$) and ($b$) confirm that WFQ efficiently enforce bandwidth reservation, under the considered conditions.

A different behaviour has though been observed with bursty QoS traffic. In real systems, burstiness may derive from internal congestion at the source host. In the case of DS, bursts can be as well originated by the overlapping of small irregularities in the micro-flows aggregated into the same traffic class. The use of PQ allows to guarantee the required delivery rate and a low end-to-end delay, as shown in Figure 8, where the performance is shown for a group of 10 routers with uniform background traffic and using QoSMIC as the QoS multicast routing protocol. However, the priority packet scheduling policy is *unfair*: it allows to guarantee only the service provided by the highest priority traffic class. This policy could be adopted for instance to implement a qualitative service such as the Assured Forwarding PHB [17, 15] in DS domains. We are performing experiments to evaluate the performance achieved by the different forwarding classes and dropping levels with this implementation. By contrast, WFQ supports different service levels by enforcing the separation amongst reserved resources. But, it can fail in providing QoS in the presence of bursts. Indeed, in this case WFQ performs a sort of *traffic shaping* in the congested routers. The QoS packets exceeding the reserved bandwidth are buffered, and as the consequence their delay increases. In highly congested nodes the QoS packets can be dropped, thus slightly decreasing the delivered rate. This is confirmed by observing the queue length: in Figure 9($a$), we report the QoS queue dynamics over 20 sec. of simulation time, for a QoS source rate of 1.9 Mbps (hence, 19 out of 20 buffer locations are reserved for QoS packets). To try to overcome this problem, we configured the routers so that the amount of reserved bandwidth is 5% greater

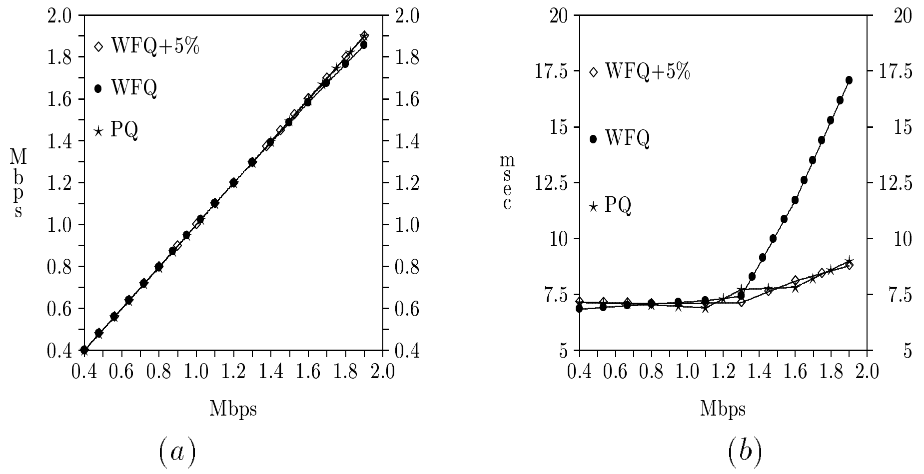



Figure 8: $(a)$ Received rate vs. offered QoS load for different packet scheduling policies, in the presence of a bursty QoS source. $(b)$ Average end-to-end delay.

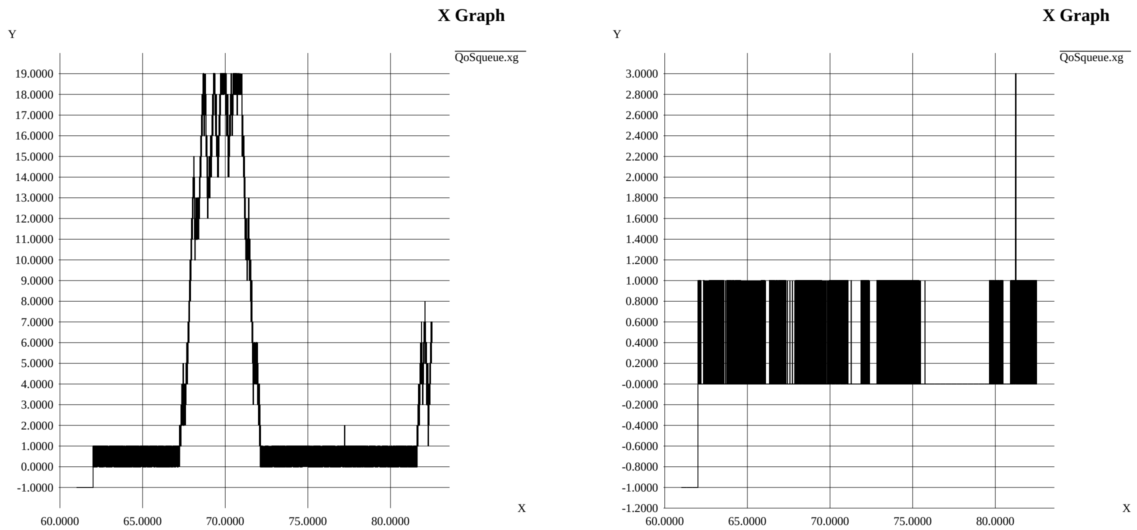
than the source rate. This case, the observed performance is comparable with that achieved with PQ (Figure 8), and the number of queued QoS packets is low (Figure 9($b$)). However, overbooking represents a waste of bandwidth. We are implementing a `shaper` module that runs in each router, to shape the out-of-profile traffic. We would like to determine whether traffic shaping in routers plenty of resources allows to reduce congestion in the downstream routers, thus allowing to deliver the requested QoS avoiding overbooking.

Our first results show that the ⟨QoSMIC, WFQ⟩ stack should allow to provide the Controlled Load service [30] in IS domains, also in the presence of burstiness. By contrast, the Guaranteed QoS service [26] may probably be supplied by performing the traffic shaping at both the source and the intermediate routers. This way, bursts are eliminated and QoS packets are immediately forwarded, thus guaranteeing no queuing delay.

To provide the Expedited Forwarding PHB [20] in DS domains, both shaping and delivery functionalities must be available. The `delivery` module annihilates the jitter, as observed when using RTP in our experiments. The `shaper` guarantees both the maximum arrival rate and the minimum departure rate by eliminating bursts. We are enhancing our framework with this module to perform further experiments and verify these observations.

# 5 Concluding remarks

This paper presents some performance evaluations and comparative analyses of different protocol stacks proposed in the literature to support QoS in the Internet. We provide an overview of the architectural issues associated to the transport and to the routing of QoS-sensitive traffic in an inter-domain network. We describe a simulation framework developed ad hoc on top of the NS-2 simulation package to perform extensive experiments with QoS-sensitive architectures and protocols. The defined framework is mainly intended to ease the deployment of new QoS-sensitive infrastructures. It may represent a common tool usable by different research groups to cooperate in order to deploy a QoS platform. We report the performance measures we carried out by using the simulation framework and combin-




$(a)$ $(b)$

Figure 9: Number of queued QoS packets vs. time (sec.), with $(a)$ the correct resource reservation; and $(b)$ 5% of bandwidth overbooking.

ing in various ways modules natively available in NS-2, available from public domain or implemented ad hoc.

The obtained results show that scheduling policies have a relevant impact on the achieved performance, depending on the way they enforce resource reservation and on their ability to cope with traffic irregularities. The results allow to characterize what protocol stack is better suitable for each service. Different traffic classes require different policies; it is still an open issue how to provide the coexistence among those policies. The results also highlight a few critical aspects of the current QoS process; the most important of them is that QoS is introducing extra complexity in the routers in terms of both time and resources consumption. From one point of view, the complexity at the routing level can represent a challenge for router manufacturers to build a new generation of high-performance, hardware-based routers. By contrast, it also requires a thorough analysis of the way to reduce or to overwhelm complexity, thus expediting the deployment of needed QoS services. One way to reduce the complexity is to privilege diff-serv's, which allow to statically define a bounded number of PHB's and to change them periodically. Dynamic creation of service classes or of int-serv flows seems unfeasible with the current technology when scaling to large amount of users. The described scenario seems to be the most affordable way to provide QoS during the technological transition towards the new generation of high speed, optical network infrastructures. At that time, a new generation of simplified QoS-capable routers will be probably designed and deployed.

The protocols proposed in the literature for the end-to-end level only partially cover the issues that have to be dealt with at this level. In particular, in the case of multicast, many open problems still exist concerning the management of heterogeneous receivers. For instance, some sort of agreement protocol could be needed when fair delay requirements have to be satisfied, to guarantee an homogeneous service to all the destinations. If on the contrary different QoS's must be supplied to different receivers, policies have to be devised to efficiently manage the many flows possibly generated as the consequence. These issues call for the design of original QOS-sensitive multicast protocols; this is one of our current objectives.

Currently, we are as well working toward the deployment of a testbed environment that will allow to perform extensive experiments with the proposed architecture in a real network, by modularily combining software tools either available from public domain or implemented ad hoc.

# References


[1] `http://www.cs.columbia.edu/~hgs/rtp/`.

[2] Aurrecoechea C., Campbell A. T., Hauw L. "A Survey of QoS Architectures". *ACM/Springer Verlag Multimedia Systems Journal, Special Issue on QoS Architecture, Vol. 6, No. 3*, pages 138–151, May 1998.

[3] Bagnall P., Briscoe R., Poppitt A. "Taxonomy of Communication Requirements for Large-scale Multicast Applications". *Internet Draft, draft-ietf-lsma-requirements-02.txt*, Nov. 1998. Work in progress.

[4] Ballardie A. "Core Based Trees (CBT version 2) Multicast Routing". *RFC2189*, Sep. 1997. Work in progress.

[5] Banerjea A., Faloutsos M., Pankaj R. "Designing QoSMIC: A Quality of Service Sensitive Multicast Internet protoCol". *Internet Draft, draft-banerjea-qosmic-00.txt*, Oct. 1998. Work in progress.

[6] Bennet J.C.R., Zhang H. "$WF^2Q$: Worst-case Fair Weighted Fair Queueing". *Proc. IEEE Infocom'96*, pages 120–128, Mar. 1996.

[7] Bernet Y., Binder J., Blake S., Carlson M., Keshav S., Davies E., Ohlman B., Verma D., Wang Z., Weiss W. "A Framework for Differentiated Services". *Internet Draft, draft-ietf-diffserv-framework-01.txt*, Oct. 1998. Work in progress.

[8] Biswas S., Izmailov R., Rajagopalan B. "A QoS-Aware Routing Framework for PIM-SM Based IP-Multicast". *Internet Draft, draft-biswas-pim-sm-qos-00.txt*, Jun. 1999. Work in progress.

[9] 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*, Oct. 1998. Work in progress.

[10] Braden R., Clark D., Shenker S. "Integrated Services in the Internet Architecture: an Overview". *RFC 1633*, Jun. 1994. Work in progress.

[11] Bradner S. "Internet Protocol Multicast Problem Statement". *Internet Draft, draft-bradner-multicast-problem-00.txt*, Sep. 1997. Work in progress.

[12] Estrin D. et al. "Protocol Independent Multicast - Sparse Mode (PIM-SM)". *RFC2362*, Jun. 1998. Work in progress.

[13] Fall K., Varadhan K. "ns Notes and Documentation". The VINT Project, Jul. 1999. `http://www-mash.CS.Berkeley.EDU/ns/`.

[14] Faloutsos M., Banerjea A., Pankaj R. "QoSMIC: Quality of Service sensitive Multicast Internet protoCol". *Proc. SIGCOMM 1998*, Sep. 1998.

[15] Firoiu V., Casati A. "Amendments to the Assured Forwarding PHB Group". *Internet Draft, draft-firoiu-diffserv-af-amend-00.txt*, Nov. 1998. Work in progress.

[16] Floyd S., Jacobson V. "Random Early Detection Gateways for Congestion Avoidance". *IEEE/ACM Transactions on Networking, Vol. 1, No. 4*, page 397, 1993.

[17] Heinanen J., Baker F., Weiss W., Wroclawski J. "Assured Forwarding PHB Group". *Internet Draft, draft-ietf-diffserv-af-03.txt*, Nov. 1998. Work in progress.

[18] Hou J., Tyan H.-Y., Wang B. "QoS Extension to CBT". *Internet Draft, draft-hou-cbt-qos-00.txt*, Feb. 1999. Work in progress.

[19] Internet2 QoS Working Group Draft. "Draft QBone Architecture". *Internet Draft, draft-i2-qbone-arch-03.txt*, Jan. 1999. Work in progress.

[20] Jacobson V., Nichols K., Poduri K. "An Expedited Forwarding PHB". *Internet Draft, draft-ietf-diffserv-phb-ef-01.txt*, Nov. 1998. Work in progress.

[21] Katevenis M., Sidiropoulos S., Courcoubetis C. "Weighted Round-Robin Cell Multiplexing in a General-Purpose ATM Switch Chip". *IEEE Journal on Selected Areas in Communications, Vol. 9, No. 8*, page 1265, 1991.

[22] Pagani E., Rossi G.P. "Architectural Requirements for the Transport of Multicast Packets with Maximum Delay Delivery Guarantees". *Technical Report 244-99, Dipartimento di Scienze dell'Informazione, Università degli Studi di Milano*, Nov. 1999.

[23] Pusateri T. "Distance Vector Multicast Routing Protocol". *Internet Draft, draft-ietf-idmr-dvmrp-v3-09.txt*, Sep. 1999. Work in progress.

[24] Schulzrinne H., Casner S., Frederick R., Jacobson V. "RTP: A Transport Protocol for Real-Time Applications". *RFC 1889*, Jan. 1996. Work in progress.

[25] Schulzrinne H., Rao A., Lanphier R. "Real Time Streaming Protocol (RTSP)". *RFC 2326*, Apr. 1998. Work in progress.

[26] Shenker S., Partridge C., Guerin R. "Specification of Guaranteed Quality of Service". *RFC 2212*, Sep. 1997. Work in progress.

[27] Shenker S., Wroclawski J. "General Characterization Parameters for Integrated Service Network Element". *RFC 2215*, Sep. 1997. Work in progress.

[28] Shenker S., Wroclawski J. "Network Element Service Specification Template". *RFC 2216*, Sep. 1997. Work in progress.

[29] Tanenbaum A. S. *Computer Networks*. Prentice-Hall, 1988.

[30] Wroclawski J. "Specification of the Controlled-Load Network Element Service". *RFC 2211*, Sep. 1997. Work in progress.

[31] Zhang L., Braden R., Berson S., Herzog S., Jamin S. "Resource reSerVation Protocol (RSVP) - Version 1 Functional Specification". *RFC 2205*, Sep. 1997. Work in progress.