Architectural Requirements for the Transport of Multicast Packets with Maximum Delay Delivery Guarantees*

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

Technical Report 244-99

Abstract

This paper describes an architectural framework for end-to-end QoS in large multi-domain networks, where int-serv and diff-serv are both deployed to provide service differentiation. The described framework is intended to ease the deployment of new QoS-sensitive infrastructures and to expedite the interoperability among etherogeneous domains.

Unicast and multicast traffics are both considered. For both we characterize the functional modules which compose the architecture and their relationships. We identify the architectural and communication issues which arise when multicast traffic traverses a multi-domain network and we show how the architecture must be modified to support that traffic under different end-to-end QoS requirements. We finally report some initial performance evaluations obtained by implementing the multicast architecture for CBR traffic in the frame of a simulation environment based on the *ns* simulation package.

This paper is the first attempt to provide a common and homogeneous architectural framework to deal with multicast communications with QoS guarantees in a multi-domain network. This is demonstrating to be very helpful to coordinate more specific analysis through simulations and to deploy a testbed environment which allows more extensive evaluations by modularily combining software objects available from public domain.

Keywords: multimedia applications, multi-domain networks, Quality-of-Service, multicast, border routing.

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

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 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 [30], RTP [23] or more recently QoSMIC [13], are just a few examples of the efforts in this direction, while *integrated services* (int-serv or IS, for short) and *differentiated services* (diff-serv or DS, for short) 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.

This paper describes 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 main advantages obtained by the proposed architecture are the following:

1. it provides a common functional framework to which the QoS initiatives can refer and which is helpful to identify duplicates and the relationship among functional modules;

2. it provides the building blocks to design the architecture of the routers operating at the border between domains adopting different service differentiation policies;

3. it is adequate to support both unicast and multicast packet flows;

4. it guarantees backward compatibility with existing protocols and services.

The described framework is intended to ease the deployment of new QoS-sensitive infrastructures and to expedite the interoperability among domains adopting different differentiation policies.

The paper is organized as follows: in section 2 we introduce the framework for the QoS problem and we present the system model and the notation adopted throughout this work. In section 3 we describe in some detail the proposed functional architecture, and we show how it can be applied to design the routers operating at the border between adjacent, etherogeneous domains. In section 4, the routing and the QoS architectures are expanded to support QoS and *multicast*-sensitive traffic. In section 5, we describe our initial implementation of the proposed architecture, and we discuss the performance measures we have obtained by means of simulation techniques.

2 Background and notation

A QoS-sensitive application specifies to the underlying service provider, the QoS communication requirements and the behaviors of the data flow it is going to generate (*traffic profile*).

So far, two main service differentiation models have been provided: *integrated services* (int-serv) [8, 29, 25, 26, 27] and *differentiated services* (diff-serv) [7, 5, 20, 15]. Int-serv's offer QoS on a per-flow basis. The protocols implementing int-serv exchange signaling messages to perform resource reservation and packet classification along the path from source to destination. 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 (or *behaviour aggregates*), 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 [20], 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.

The interconnected system we will consider in the following is composed of a set of networks connected by routers (Figure 1(a)). The hosts in the networks may run applications that require a QoS transmission service. Routers allow the forwarding of messages through

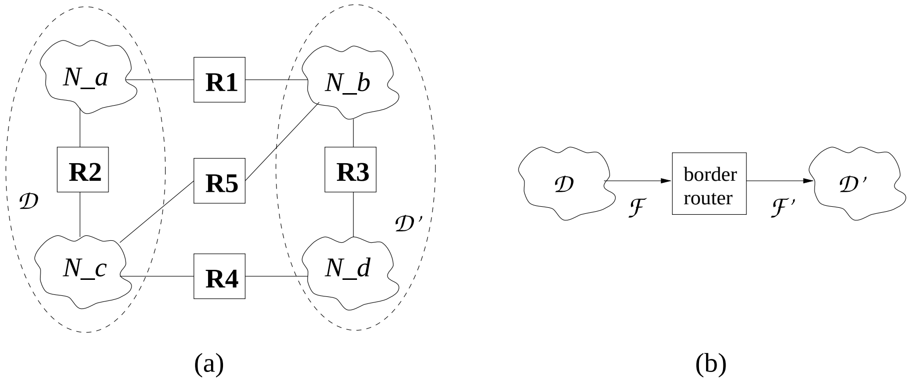

Figure 1: The system model.

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.

According to the notation adopted for diff-serv [7], we denote as *ingress* routers those routers connected to a network involving hosts being source of QoS traffic. Similarily, *egress* routers are the routers connected to a network including a host which is destination of QoS traffic. In Figure 1(a), if a host h_a in the network N_a generates a data flow to a host h_d in the network N_d, and the path from N_a to N_d is through the routers R_1 and R_3, then R_1 is the ingress router for that flow while R_3 is the egress router.

We call *domain*, a set of contiguous networks which operate with a common set of service differentiation models, provisioning policy and service definition. The system shown in Figure 1(a) considers two network domains, $\mathcal{D}$ and $\mathcal{D}'$. For instance, the networks N_a and N_c adopt the int-serv provisioning policy, while the networks N_b and N_d adopt the diff-serv provisioning policy and the same class definition. We denote as *border* routers those routers connecting contiguous network domains. In order to guarantee the requested end-to-end QoS (Figure 1(b)), the *border* routers (R_1, R_4 and R_5 in Figure 1(a)) are responsible of mapping the service provisioning and forwarding policy $\mathcal{F}$ to which a flow is subject in one domain into the model and policy $\mathcal{F}'$ which have to be applied in the adjacent domain. For instance, $\mathcal{D}$ might adopt int-serv, while $\mathcal{D}'$ might adopt diff-serv; or, both $\mathcal{D}$ and $\mathcal{D}'$ adopt diff-serv, but $\mathcal{D}$ provides best-effort and *premium* services [21, 19], while $\mathcal{D}'$ provides best-effort and *assured* services [16, 14]. The translation policy depends on the agreement between the adjacent domains.

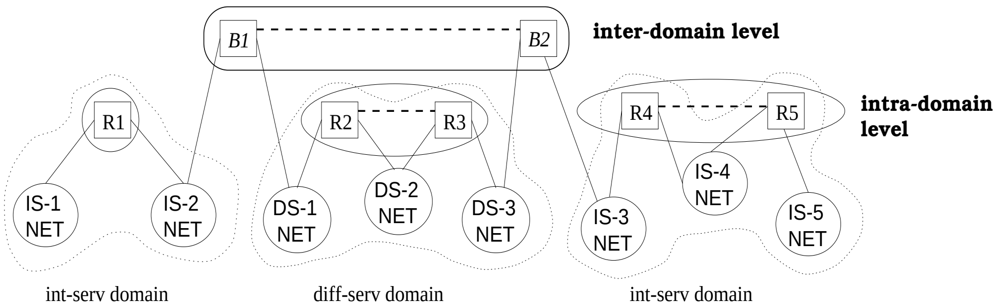

Figure 2: Routing architecture of QoS-capable routers.

A three layers architecture (Figure 2) describes how the routing is performed within the considered interconnected system [13]. At the lower layer, routers perform packet routing among QoS-capable routers within a single network. Intra-domain routers allow the routing among a set of homogeneous networks. Border routers operate at the inter-domain level and perform routing among domains. All these routing functions are performed through routing tables which are maintained at each different functional layer. We assume that the routing at the identified layers of the routing architecture is performed among QoS-capable routers [30], [8].

3 QoS functional architecture

A functional architecture can be devised to locate the functional modules needed to provide QoS over an IP-based network. Their 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 [28]. 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.

In this section, we firstly discuss in some detail the services provided by each level of the architecture. Secondly, we presents the overall architecture of a border router.

3.1 QoS Architecture

The overall functional organization of a QoS-capable element of a network is described by the QoS architecture of Figure 3. 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 3, bold lines are used to specify the control flow, which describes the *set-up phase*, while thin lines specify the data flow, which describes the *data transfer phase*. Dashed lines indicate intra-layer control/data flows which will be detailed in the next sections.

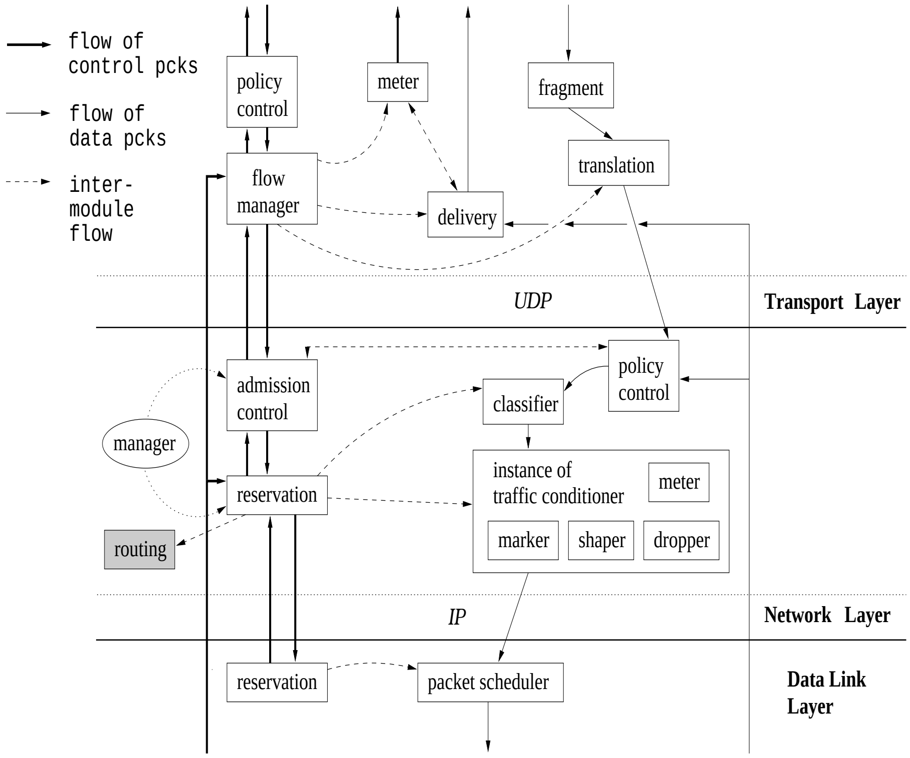

Figure 3: Functional architecture of a QoS-capable router.

The functional modules that belong to this QoS architecture can be grouped in four main classes, spanning the layers of Figure 3:

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 of a QoS flow.

traffic management class: this class groups the functions which define the QoS traffic management policy.

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.

security class: this class includes functions which allow the authentication of an application, access control, accounting and security management.

In the following subsections we discuss in detail those service classes.

3.1.1 System setting

The set of modules whose functionalities fall into this class provide the following services:

multimedia application support: Since applications are likely to be unaware of the network internals, a *lieson* service should be provided between host and network functions. The `flow manager` module provides an abstraction of the underlying QoS implementation. It translates the independent application specification into the format which is suitable for the network layer (e.g., a `TSpec` and a `RSpec` [26, 27] in the case of int-serv's, or a Per-Hop-Behavior (PHB) in the case of diff-serv's).

resource reservation: Under specific QoS requirements, end-to-end resource allocation must be performed. For instance, when considering a CBR traffic, the `flow manager` performs end-to-end signalling to ensure that CBR guarantees are provided at the destination system, i.e. availability of the needed buffers and delivery at a constant rate [22].
At the network layer, the accounting of both the reserved and the available resources must be maintained (`admission control` module). Whenever a request for a new QoS flow or behaviour aggregate is received, the accounting allows to decide whether or not it can be satisfied. If this is the case, the appropriate reservation is performed (`resource reservation` module). By interpreting either the `TSpec`, in case of IS requests, or the traffic profile and class specified in the *Reservation Request Object* (RRO) packets [18], in case of DS requests, the `resource reservation` module reserves buffer space at both the network and data link layers. `Admission control` and `resource reservation` modules could actually be fused into a single module. They are separated here to properly model the nature of the RSVP protocol [30].

module configuration: The modules that process the QoS traffic during the data transfer phase have to be appropriately configured during the set-up phase, so that they correctly manage the packets belonging to the different flows. At the end-to-end layer, the configuration concerns the modules responsible for fragmenting end encoding the generated messages (`translation`) in the source node, and for delivering the received packets according to the required QoS (`delivery` module) at the destinations. The configuration is performed by the `flow manager` module.
At the network layer, the modules responsible for forwarding the data flows must be configured, in order to ensure the proper processing of the data packets during the data transfer phase. This task is carried out by the `resource reservation` module, that also generates control information to configure the underlying `packet scheduler` so that it implements the appropriate forwarding policy for the different classes of traffic.

The preliminary configuration of the `admission control` and `resource reservation` modules must be performed as well, by the `manager` module. In the border routers, the configuration includes the translation policy to support interoperability among different domains. The configuration can be performed manually; in this case, the `manager` is the network administrator.

3.1.2 Traffic management

In this section, we consider the set of modules which operate during the data transfer phase, to process the data packets so that the required end-to-end QoS is achieved. To this purpose, we can characterize the following services:

management of the generated data: the application messages have to be fragmented into transfer units which possibly match the size of network packets (`fragmentation` module). When the packets flow through different sub-networks, further fragmentation at the network layer can create problems to ensure end-to-end QoS guarantees as the consequence of the QoS-information loss over small fragments. Although current RFCs do not provide a thorough analysis of the problem, two solutions can be in practice identified: the first requires that information about the fragmentation is propagated along the path from the source to the destination [7]; otherwise, the fragmentation at this layer generates fragments as large as the smallest fragment of the traversed networks [30].
The data packets produced by the fragmentation must be marked in order to be processed according to the specified requirements (`translator`). The `translator` module is also in charge of dinamically changing the data encoding (audio/video encoding) to meet the underlying available resources or mapping multicast addressing onto unicast addressing scheme [23].

packet forwarding: packets must be handled in different ways depending on the end-to-end QoS specifications of the flows to which they belong. The data packets in a traffic stream have to be selected (`classifier` module) on the base of the content of one or more fields of the packet header. The portion of the header to which the matching function has to be applied is provided during the set-up phase by the configuration services. Each data flow (in the case of int-serv) or each class of service (in the case of diff-serv) characterized by means of the classification is separately managed, by different instances of the `traffic conditioner` module. Before forwarding the packets, traffic bursts may be either re-shaped according to the traffic profile of the service class to which packets belong (`shaper` sub-module), or demoted to a worse service class (`marker` sub-module). Otherwise, out-of-profile packets may be dropped (`dropper` sub-module), if all else fails. In the border routers, re-marking is also applied to the packets crossing a domain boundary. The re-marking policy may be either described in the specification of the service, or agreed among the administrators of the adjacent network domains.
Then, packets are forwarded to the output links. The decision about the order in which packets are forwarded is taken according to the class of traffic to which packets belong (`packet scheduler` module). Packets can be ordered in the output queues according to a FIFO, priority or weighted fair policy [19].

delivery: the messages are delivered to the upper layer (`delivery` module) according to the specified QoS requirements. When dealing with a multimedia flow, all the media streams must be kept synchronized.

3.1.3 Metering

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 end-to-end parameters evaluation is performed by the `meter` module at the end-to-end layer. To this purpose, 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 [23]. The results can be also used to dinamically re-configure some of the functional modules (e.g., the `translation` module) 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 also used at the network layer (`meter` sub-module) 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.

3.1.4 Security

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, which are provided by the `policy control` modules. 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, the `policy control` module checks the correctness of a specified portion of the packet header and its coherence with the QoS requested at the set-up time.

3.2 Border router architecture

In this section, we show how the proposed architecture can be applied to provide interoperability among different domains. To this purpose, we consider the border router architecture and we describe the intra-module communication flows in the case of a QoS traffic flowing through the IS-DS-IS domain configuration (Figure 4(a)) and through the DS-IS-DS configuration (Figure 4(b)). We assume that resource reservation in the int-serv domains is performed by using RSVP, and that both the diff-serv domains in Figure 4(b) use a static resource reservation policy with the same reconfiguration rate. Static reservation in a diff-serv domain means that the ISP a-priori decides the amount of resources to reserve for each

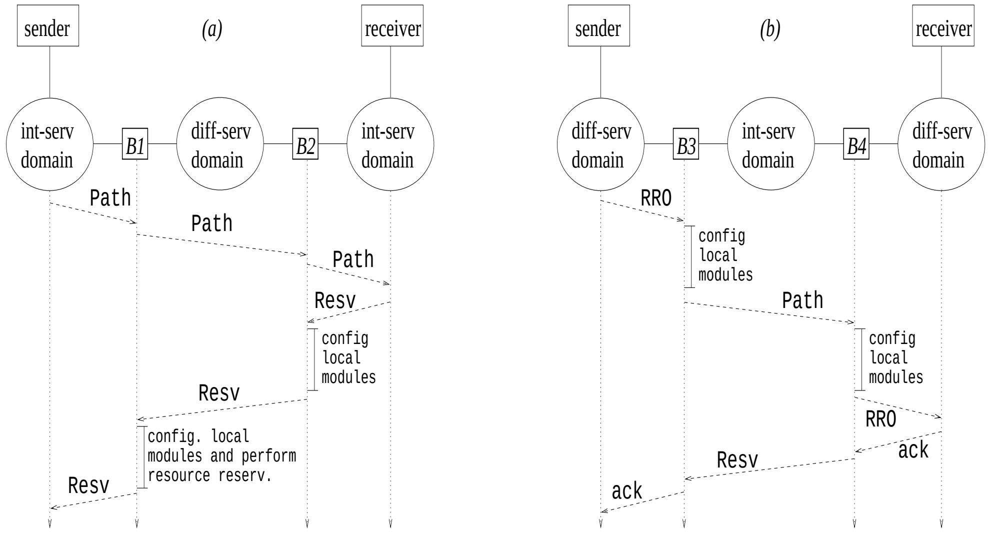

Figure 4: Multi-domain configurations and resource reservation procedures.

PHB by running the resource reservation setup procedure [18]. Periodically (e.g. monthly, quarterly) reservations are re-configured on the base, for instance, of new user requests and of the traffic pattern observed so far.

In the following of this section, we describe the intra-module interactions for border routers in the configurations of Figure 4. For sake of simplicity, we provide the separated description of the flows concerning the control (i.e., the set-up) phase and the data (i.e., the data transfer) phase. In line with that, in Figure 5 we show the control phase for the border routers $B1$ and $B2$ (Figure $4(a)$), and in Figure 6 we show the data transfer phase for the same pair of border routers.

The structure of a generic border router can be easily obtained by merging all the flow diagrams described in the following.

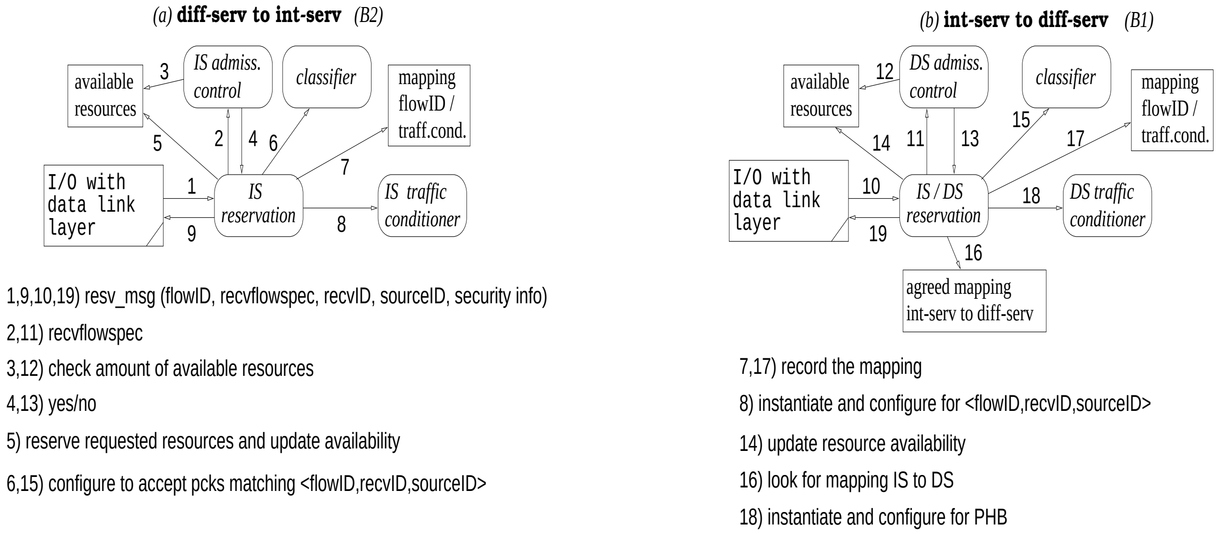

Figure 5: Border router control flows for the configuration IS-DS-IS.

When we consider the configuration of Figure 4(a), the DS domain provides a transparent tunnel between the two IS domains, where a packet belonging to a given IS flow is incapsulated into a DS packet. The modules of the border routers are activated in the sequence $B2$, $B1$, by the backward flow of the `Resv` packet generated by the destination upon receiving the `Path` packet, Figure 4(a). When the border router $B2$ receives the `Resv-msg` (data flow 1), the `reservation` module contacts the `admission control` module which checks whether enough resources are locally available to support the received flow specification (2)(3). Upon receiving the reply (4), the `reservation` module updates the resource availability (5) in the event of sufficient resources. At this point, the `reservation` module is ready to configure the router to process the new incoming data packet flow. To this purpose, it configures the `classifier` (6), instantiates the `traffic conditioner` for that flow (8) and records the association between flow-identifier and `traffic conditioner` into a local table (7) which will be read, during the data transfer phase, by the `classifier` in order to direct the incoming packets to the proper `traffic conditioner`. The `classifier` is instructed to select packets according to a matching rule applied to the set of packet fields ⟨flowID, recvID, sourceID⟩, i.e., the `classifier` is of type *MF (Multi-Field) classifier* [7]. The `marker` module of the `traffic conditioner` is configured to eliminate the DS-compliant envelope from the data packets in order to generate IS packets. At the end of this sequence of actions, the

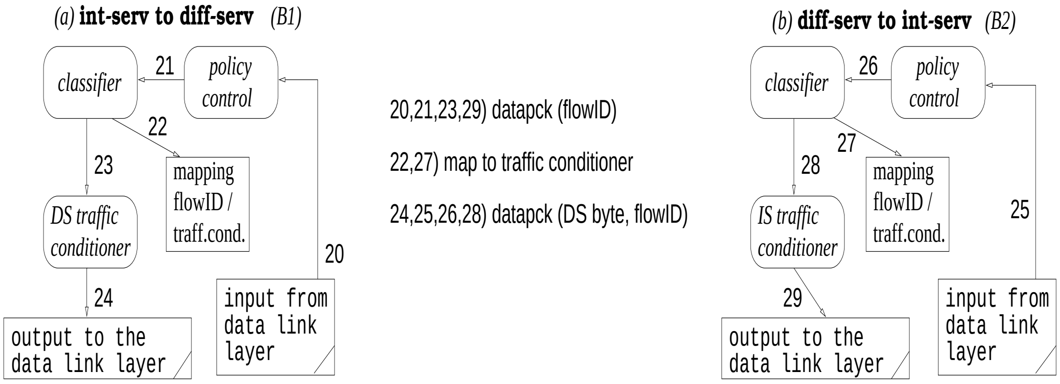

Figure 6: Border router data flows for the configuration IS-DS-IS.

`Resv-msg` is forwarded back to the data source (9).

The control flows of border router $B1$, Figure 5(b), and those of $B2$ are very alike. However, since $B1$ has the responsibility of activating the transparent tunnel through the DS domain, the local `reservation` module must be able to map the traffic profile of the incoming IS packets into the proper PHB on the DS tunnel. To this purpose, the mapping information are available in a local table. The table is accessed by the `reservation` module (16) and is maintained by the `manager` module which records the mapping agreements defined between adjacent domains. The update rate of this table can be considered very low.

Figure 6 describes the data transfer phase for the configuration in Figure 4(a). When $B1$ receives a data packet (20), the `policy control` module applies the policing rules (21), and the `classifier` uses the mapping information (22) to direct the data packet to the appropriate `traffic conditioner` (23). The `marker` adds to the data packet header the DS byte, associated to the specified PHB, and forwards it to the data link layer (24). When the data packet is received by $B2$ (25), the `classifier` steers it to the proper `traffic conditioner` (28) by matching the flow information (27) and by ignoring the DS byte. The `traffic conditioner` eliminates the DS byte from the packet header and forwards it to the data link layer (29).

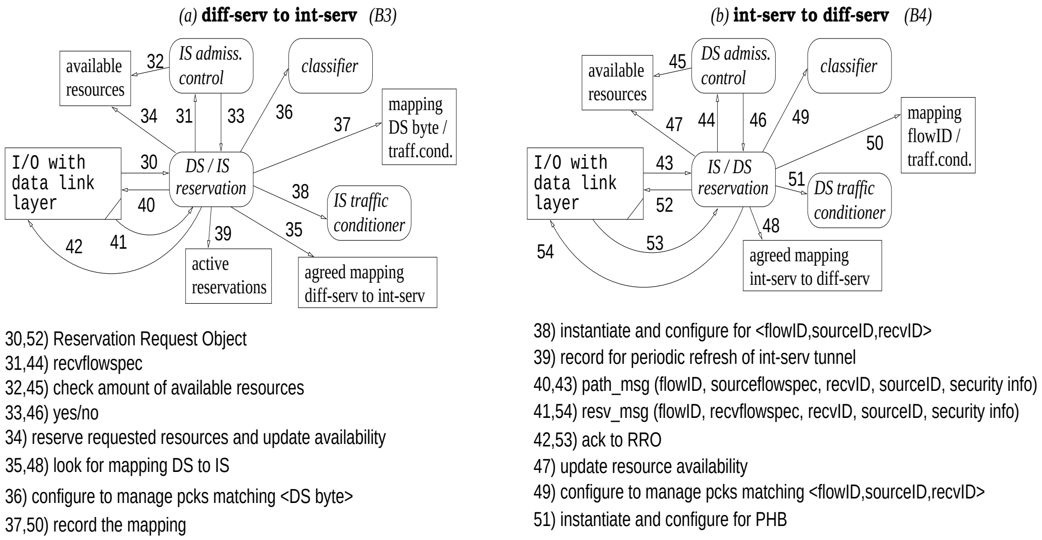

Figure 7: Border router control flows for the configuration DS-IS-DS.

Let us now consider the configuration DS-IS-DS of Figure 4(b). In this case, the IS domain provides a transparent tunnel between the two DS domains and the modules of the border routers are activated in the sequence $B3$, $B4$, by the RRO [18] which initiates the resource reservation procedure at the DS side. The control flows of Figure 7(a) and (b) can be easily followed by adopting the same arguments which have been introduced to describe the flows of Figure 5. The border router $B3$ between the DS and IS domains only requires an extra information not needed when bordering IS to DS. In fact, the IS domain is responsible of maintaining IS flows, henceforth called *IS-tunnel-flows*, which encapsulate the assigned DS PHB. As the consequence of the fact that DS PHB are nearly statically defined, their life time is much longer than the flows that RSVP maintains according to a soft state maintenance policy. To allow that all the IS-tunnel-flows can last longer than the RSVP refresh rate, the `reservation` module maintains, into the local table 'active reservations', the list of the currently active IS-tunnel-flows (39). The control flows (41) and (42) in Figure 7(a) ((53) and (54) in Figure 7(b)), describe the reply flow to the packets `Path` and RRO. Ack and `Resv` packets do not require processing within the border router.

The data transfer phase has the same sequence of actions described in Figure 6. In particular, the data flow for $B3$ can be derived from the data flow of $B2$, while the data flow for $B4$ from the data flow of $B1$.

4 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 [9], 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 [11] and CBT [3]). So far, very few proposals have been presented to provide multicast QoS (e.g. RSVP [30], RTP [23], QoSMIC [13]). All these proposals consider the multicast within a single domain. Inter- and intra-domain communications are not considered in details.

Scope of this section is to expand the architecture introduced in section 3 to consider the case of multicast traffic and to provide a preliminary understanding of the architecture of a QoS and multicast capable border router. The multicast traffic slightly modifies both

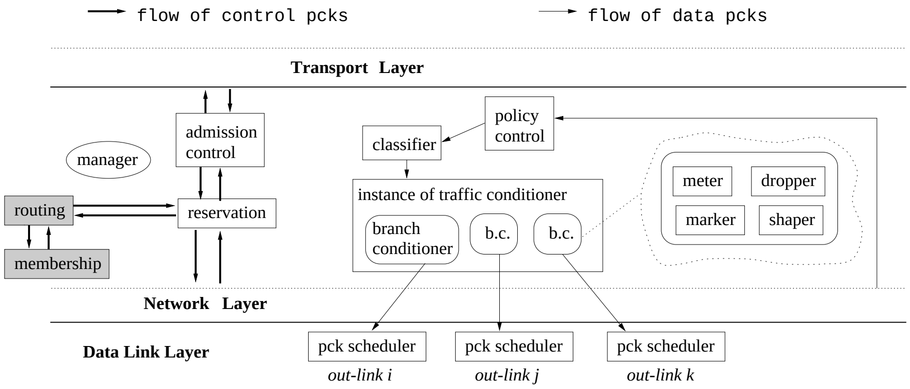

Figure 8: QoS multicast functional architecture.

the end-to-end and the network layer modules of the architecture in Figure 3. The changes to the end-to-end layer are discussed in section 4.1. In Figure 8, we show the functional architecture of the network layer of a generic QoS capable router which supports multicast traffic. The functional modules of Figure 8 are those of Figure 3 with a few minimal changes:

membership: the `membership` module has been introduced to manage the group membership.

routing: the `routing` module performs multicast routing which is supposed to maintain a tree-based multicast infrastructure among the QoS/multicast-capable routers. The membership and the routing modules are both replicated at the different layers of the routing stack, as described by the following example.

traffic conditioning: a given routing entity belonging to a multicast tree infrastructure at any level of the routing stack, may be either a leaf router or be connected to one or more downstream entities through downstream links (i.e. it is the parent of one or more sub-trees). At the inter-domain level, the downstream links are described by the traversed transit domains, at the intra-domain level by the traversed QoS-sensitive networks, and so on, to terminate with the set of physical links which transport the multicast packets. As the consequence of these arguments, a border router must instantiate one `traffic conditioner` module for each downstream transit domain. Further, each instance of the `traffic conditioner` must apply the same conditioning policy to all the physical links, which are identified at the end of the routing process within a single transit domain. That means that a copy of the same set of modules, `marker`, `shaper`, `meter` and `dropper`, must operate over the output links which carry a multicast flow. We denote this set of modules with the term `branch conditioner`; a `traffic conditioner` may contain as many `branch conditioners` as many downstream links. The following example will clarify the above arguments.

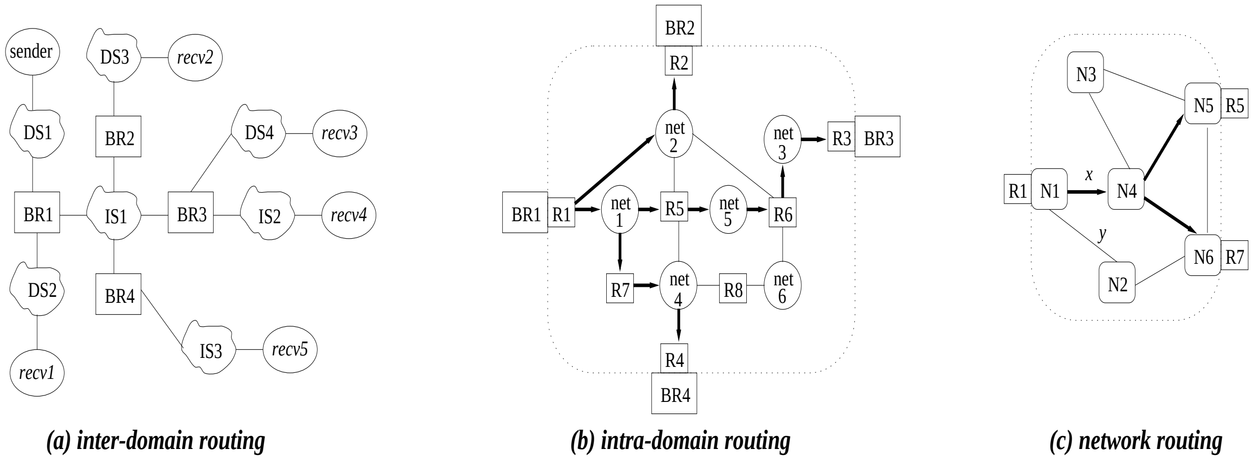

Figure 9: Example of multi-domain routing.

The Figure 9 describes the case of a multicast sensitive traffic with destinations $recv1$, $recv2$, $recv3$, $recv4$ and $recv5$. It can be useful to understand how the inter-domain multicast routing can be obtained and to describe the overall architecture of a border router. Figure 9(a) shows the network view at the inter-domain routing for the border router BR1. BR1 sees $recv1$ one-hop away, and $recv2$, $recv3$, $recv4$, $recv5$ two hops through BR2, BR3, BR4 respectively. DS2 and IS1 are the downstream transit domains of BR1. Let us detail IS1. To achieve optimal resource utilization within the IS1 domain, a multicast infrastructure should be created at the intra-domain level of BR1. To this purpose, BR1 asks the underlying routing level, let us call it R1, to deploy a multicast infrastructure connecting R1, R2, R3, R4. That means to create a group G with membership {R1,R2, R3,R4}. G can be created by using the (intra-domain) membership protocol, while the (intra-domain) multicast routing protocol is used to create the tree of Figure 9(b), bold lines. In this infrastructure, R5, R6 and R7 are *forwarder routers*, i.e. they belong to the multicast tree but they are not members of G. R1 has two downstream transit networks (let us assume that this is the last nested routing level), net1 and net2. The inner topology of net1 is detailed in Figure 9(c). The above process is recursively applied at this level. R1 asks the underlying routing level, let us call it N1, to deploy a multicast infrastructure connecting N5 and N6. That means to create a group g with membership {N5,N6}. g can be created by using the (network-domain) membership protocol, while the (network-domain) multicast routing protocol is used to create the tree of Figure 9(c), bold lines.

It is now possible to derive the routing architecture for the border router BR1, Figure 10, which shows the use of the protocol pair, membership and multicast routing, at the different layers of the routing stack to ensure the multicast routing through QoS-capable routers.

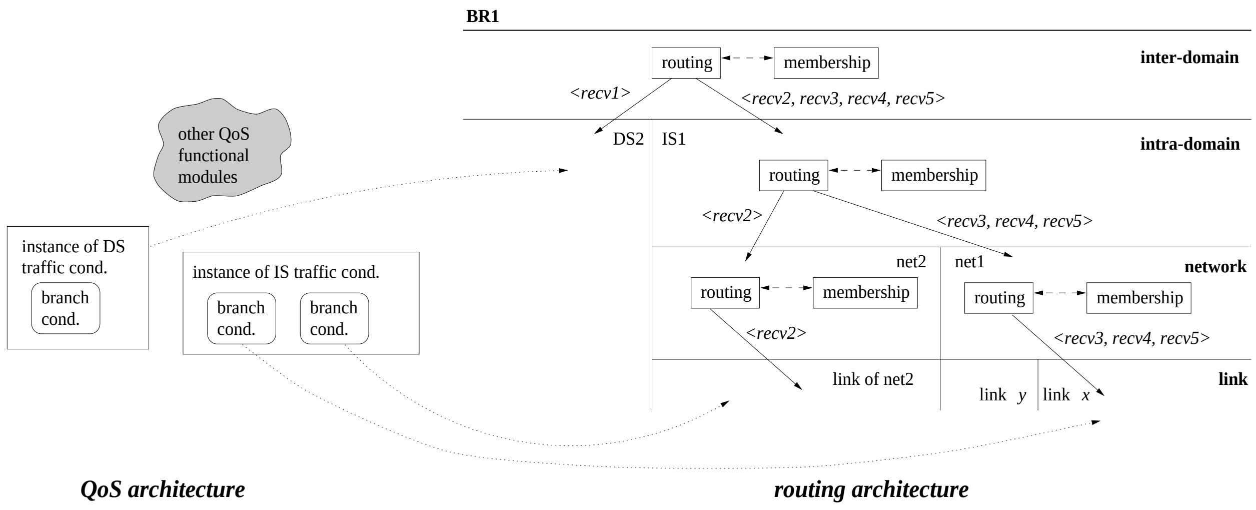

Figure 10: The QoS and routing architectures of BR1.

It should be noted that IGMP [10] cannot be used to provide the membership at the different layers. In fact, to create a group, IGMP uses a destination driven policy which requires that the group members are group aware. By contrast, to create groups G and g it is necessary to adopt a source driven policy; in fact, in G, only R1 knows the group and its membership.

Figure 10 also describes the QoS architecture for the multicast traffic. In line with the above considerations, an instance of `traffic conditioner` has been activated for the branch through DS2 and one for the branch through IS1. The latter instance is composed of two `branch conditioners`: one for link x of net1 and one for the link over net2.

4.1 End-to-end multicast control

To support the multicast traffic, the end-to-end modules are required to be group-aware, and must compensate the dishomogeneity among different destinations. In fact, the `meter` module of the source station interacts with the `meter`s within the group of destinations; it gathers QoS parameters from the destinations and notifies the measurements to the application, which decides according to application-dependent arguments. Similarily, the `flow manager` acts upon the group of `delivery` modules by performing the proper tuning according to the QoS requirements. Some problems arise when *fair delay* requirements [2] are to be satisfied. In this case, the source `flow manager` communicates with the peer modules of the destinations, that configure the local `delivery` module to deliver the data after the specified delay as estimated by the slowest destination(s). Moreover, when the multicast tree infrastructure has branches with different resource availability, the intermediate routers may resort to utilize different data encodings/compressions over different tree branches. As the consequence, the `translator` and the `traffic conditioner` modules are likely to be more complex.

Finally, 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.

5 Preliminary simulation results

The use of simulations can be helpful to understand how the different functional modules of the described architecture impact on the performances of the QoS parameters. Starting from the *ns* simulation package [12], we have developed a modular QoS 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 [2] under different traffic sources. We are currently running simulations for CBR traffic whose preliminary results are reported in this section. A more thourough evaluation will be described in the final version of the paper.

The architecture under test is described in figure 11, where the `routing` and `reservation` modules are obtained by an implementation of the QoSMIC protocol [13, 4], which has been choosen among others [17, 6] for its simplicity and flexibility. QoSMIC builds a shared tree

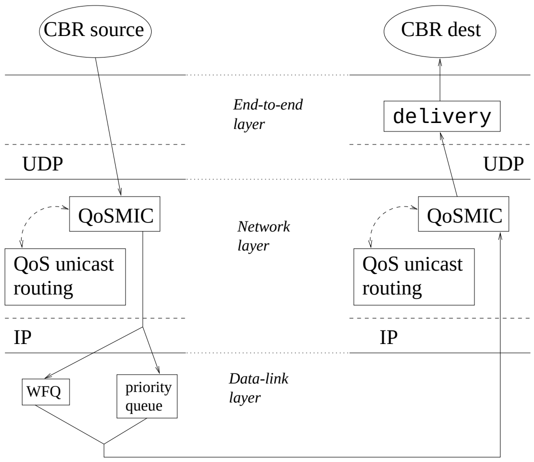

Figure 11: Layout of the implemented architecture.

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. 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 according to some *quality-of-path* metrics. QoSMIC can consider only one metric at a time; it requires an underlying QoS unicast routing protocol that finds "good" paths according to that metric. 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 11, 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 `packet scheduler` module can be alternatively implemented through a *weighted fair queue* (WFQ) or a *priority queue*. So far, the `delivery` module delivers the packets as soon as they are received.

In our experiments we are considering 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 is uniformly distributed all over the network and the average use of the bandwidth resources is 33%. As the consequence, the contention between QoS and best effort traffic initiates when R generates traffic at about 1.3 Mbps.

In Figure 12(a) we report the average rate of the QoS traffic observed at the destination nodes for different group cardinalities and offered QoS load, and by initially using the WFQ. In Figure 12(b) we show the rate loss percentage under the same conditions. The light loss

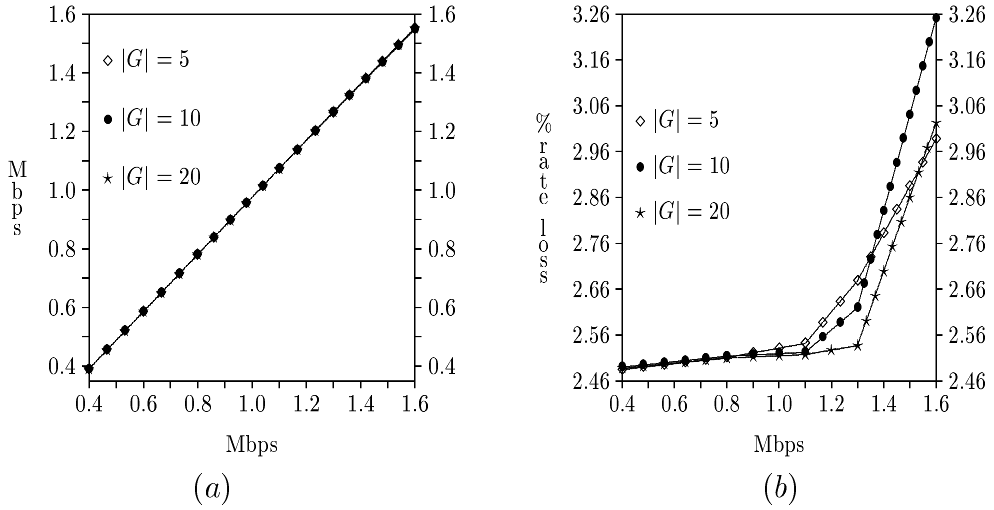

Figure 12: (a) Received rate vs. offered QoS load for different group cardinalities. (b) Percentage of rate loss.

is due to the higher end-to-end delay that the QoS packets experiment when competing with best effort packets. When the transmission rate grows beyond 1.2 - 1.3 Mbps there is a higher probability of generating bursts, and, under these conditions, to guarantee the fairness the WFQ enforces the per-flow specified bounds over the amount of reserved resources. As the consequence, packet bursts are queued, thus increasing the observed average delay at the destinations, as shown in Figure 13(a). We have proved, by modifying the QoSMIC implementation, that the problem does not arise from the QoSMIC limitation of considering a single metric.

To prove these arguments, we used a priority queue under the same conditions; the observed delay is shown in Figure 13(b). In this case, the cost of having stable delay and no rate loss for the QoS traffic is payed by the best effort packets which are progressively thrown out of the network. By increasing the group cardinality, the average delay decreases because the receivers are more densely located around the root. These first simulations confirm that WFQ fairly assigns bandwidth to different flows by reshaping the packet bursts according to the specified per flow requirements, while the use of priority queues unfairly privileges the flows with the highest priority. As the consequence, priority is not adequate in a multi-flow environment, and the attitude of WFQ to delay or discard the peak packets can be compensated only by overbooking the bandwidth.

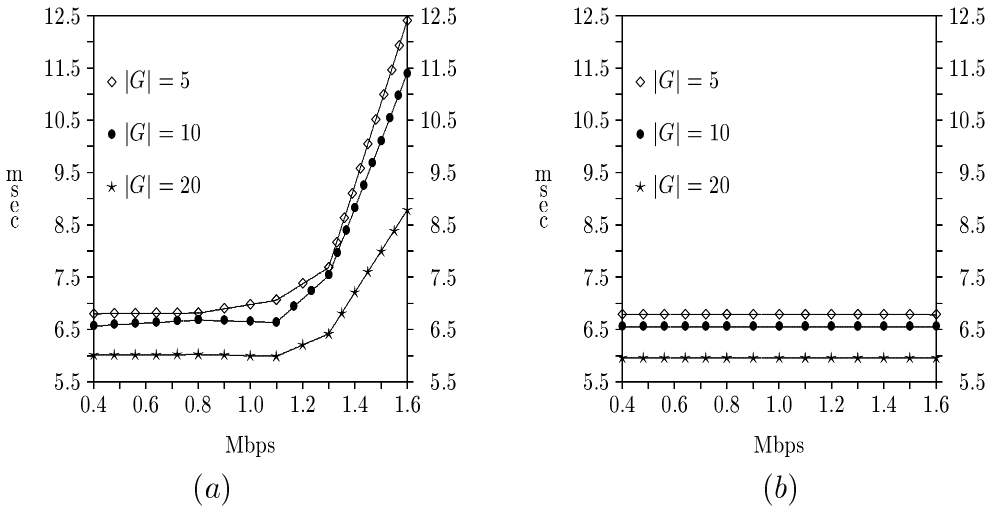

Figure 13: Average end-to-end delay vs. offered QoS load for different group cardinalities: (a) using WFQ's; (b) using priority queues.

Moreover, our first results clearly show that the bandwidth reservation alone is unable to totally control the other QoS parameters. In Figure 14 the *fair delay* is reported. Fair delay is defined as the difference between the average delays observed by the fastest and by the slowest destinations in G. Fair delay is negatively affected by the offered load (Figure 14(a)) and by the distance in hops from the source to the pair of the nearest and the farest destinations. By using priority queues (Figure 14(b)) the fair delay only depends on the topology, and this becomes more evident when a large group is considered, because the distance between the nearest and the farest destinations is likely to increase.

Similar arguments apply to the *jitter*. The jitter is the variance of the packet interarrival time at a given destination. In Figure 15 we report the jitter averaged on all the group destinations. When WFQ's are used (Figure 15(a)), the behaviour of the average jitter is chaotic. If, by contrast, the priority queues are used (Figure 15(b)) the jitter is almost zero.

The fair delay and the jitter measures confirm that the `packet scheduler` plays a relevant role. In fact, when WFQ's are adopted, the `delivery` module is required to deliver packets at a constant bit rate, thus compensating the jitter behaviour. In order to control the fair delay, by using both the scheduling policies, a cooperation amongst the `delivery` modules of the destinations in G is required that forces all of them to deliver the packets after the delay observed by the slowest receiver. In this case, the `flow manager` and `meter` modules of the traffic source should cooperate to properly configure the destination `delivery` modules. We are currently implementing these functional modules in the frame of our simulation environment, by using a simplified version of RTP and RTCP [23], to perform further experiments.

6 Compatibility with existing protocols

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 [23, 30, 24,

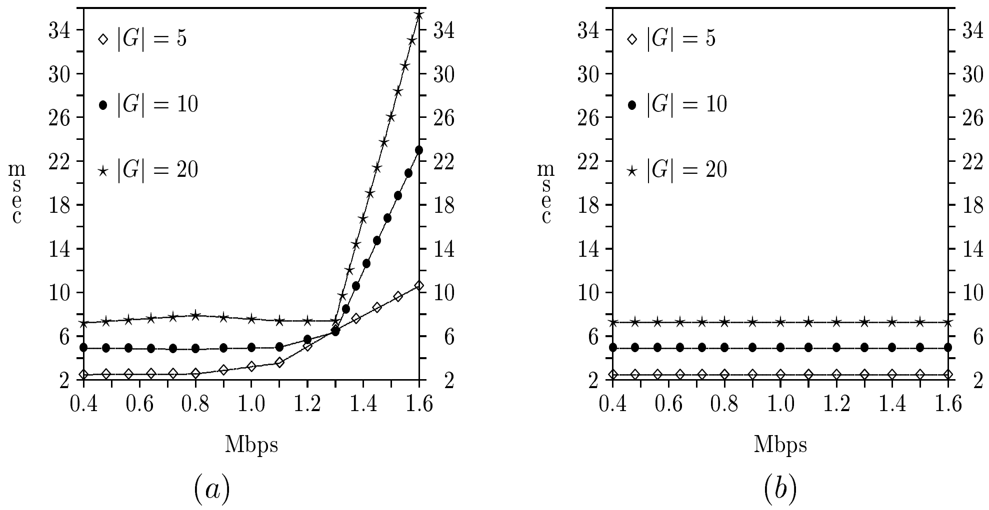

Figure 14: Fair delay vs. offered QoS load for different group cardinalities: (a) using WFQ's; (b) using priority queues.

4, 17]. 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 [1]. In Figure 16 we show how some of the existing protocols fit into the proposed architecture. The current version of RTP [23] covers the functions of the modules `policy control`, `delivery`, `translator` and `meter` (RTCP). If RTSP [24] is used on top of RTP, the modules `delivery` and `flow manager` must be partially duplicated to manage the multimedia flows as aggregate of separate media streams. The protocol RSVP [30] partially covers the functions of the `reservation` module. The Bandwidth Broker described in [21] covers the `reservation` and the `admission control` modules. The QoSMIC protocol [13, 4], as well as the CBT and the PIM-SM protocols enhanced to support QoS [17, 6], covers the `routing` and the `reservation` modules.

Figure 3 shows a complex architecture at the network layer. Complexity is bringing together a high computational cost which is mainly concentrated in the `reservation`, `classifier` and `policy control` modules. The former is running during the set-up phase and is time and CPU consuming. The latters are running during the data transfer phase and are expensive too, but a part of their functions can be optimized further and the classification process, which is the most time consuming, is suitable to be realized by hardware components.

7 Concluding remarks

This paper provides an insight view of the architectural issues associated to the transport and to the routing of QoS-sensitive traffic in an inter-domain network. The defined common framework is mainly intended to ease the deployment of new QoS-sensitive infrastructures and to expedite the interoperability among etherogeneous domains. However, it also highlights 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

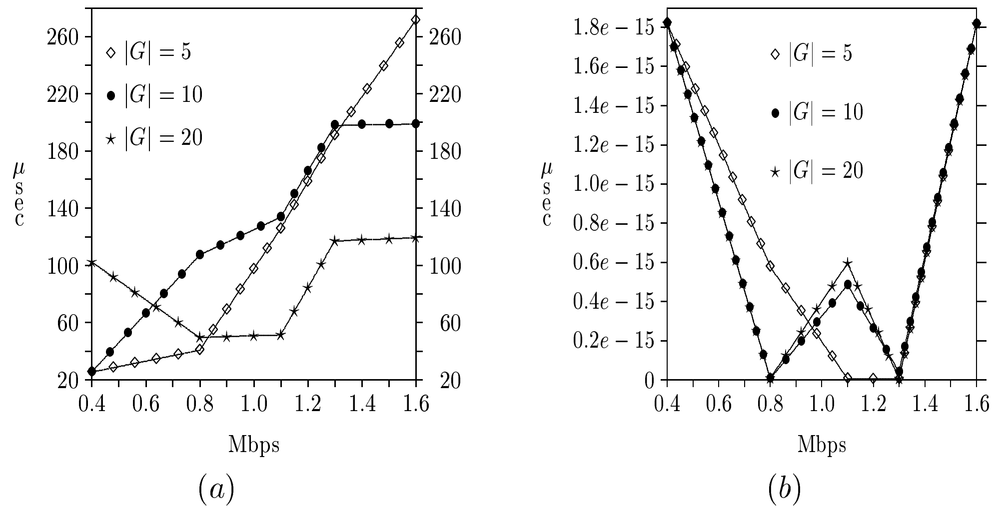

Figure 15: Jitter vs. offered QoS load for different group cardinalities: (a) using WFQ's; (b) using priority queues.

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.

In the frame of this open debate, this work provides the architectural support to the following arguments:

1. the complexity is highly concentrated during the set-up phase. This is a hint to privilege diff-serv, which allows to statically define PHB and to change them periodically (e.g.,monthly). Dynamic creation of service classes or of int-serv flows seems unfeasible with the current technology when scaling to large amount of users in a multi-domain network;

2. under the above assumptions, the inter- and intra-domain communication can be simplified as the consequence of the agreements among adjacent and co-working ISPs. The agreements allow to ease inter-domain routing and, in case of multicast, the tree construction and a QoS platform could actually be deployed through high-performance routers with hardware-based classification and fast access mechanism to table entries.

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.

Currently, we are working toward the deployment of a testbed environment that will allow to perform extensive experiments with the proposed architecture, by modularily combining software tools available from public domain, such as some of those mentioned in section 6.

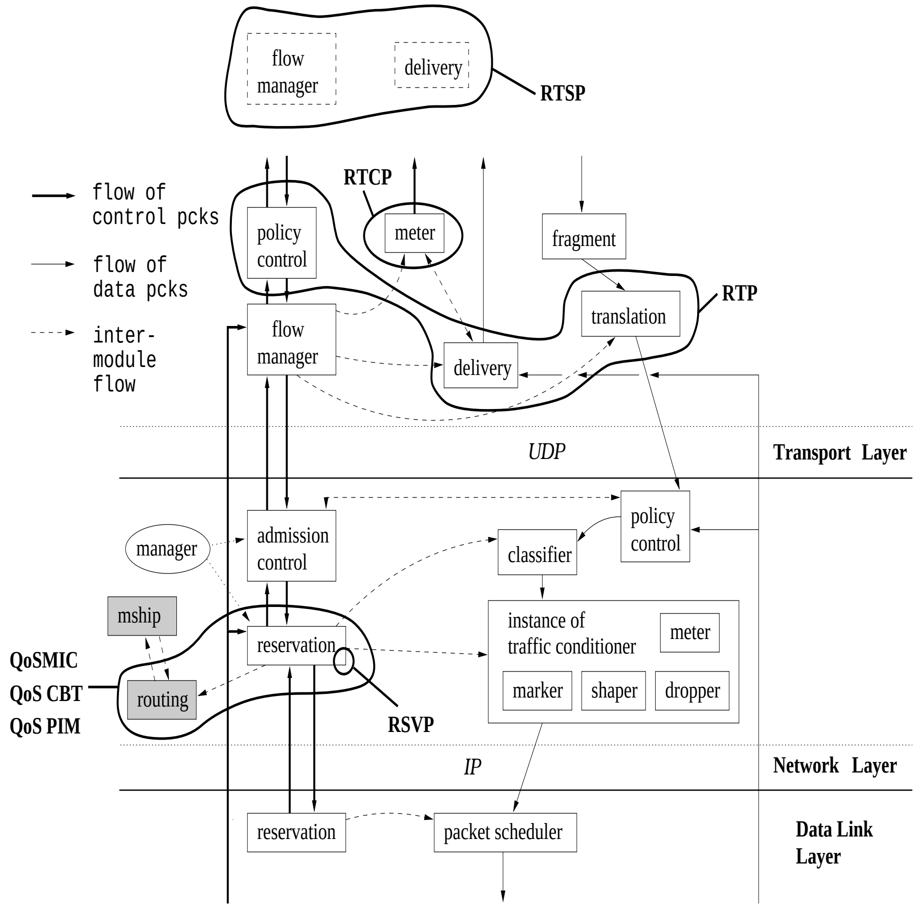

Figure 16: Mapping of some available QoS protocols onto the architecture.

References

[1] 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.

[2] 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.

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

[4] 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.

[5] 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.

[6] Biswas S., Izmailov R., Rajagopalan B. "A QoS-Aware Routing Framework for PIM-SM Based IP-Multicast". *Proc. GLOBECOM 2000*, Nov. 2000. (Also appeared as an Internet Draft, draft-biswas-pim-sm-qos-00.txt. Jun. 1999. Work in progress).

[7] Blake S., Black D., Carlson M., Davies E., Wang Z., Weiss W. "An Architecture for Differentiated Services". *RFC 2475*, Dec. 1998. Work in progress.

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

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

[10] Cain B., Deering S., Thyagarajan A. "Internet Group Management Protocol, Version 3". *Internet Draft, draft-ietf-idmr-igmp-v3-01.txt*, Feb. 1999. Work in progress.

[11] Estrin D. et al. "Protocol Independent Multicast - Sparse Mode (PIM-SM): Protocol Specification". *RFC 2362*, Jun. 1998. Work in progress.

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

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

[14] 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.

[15] Ford P. "Integrated Services over Differentiated Services". *Internet Draft, draft-ford-issll-diff-svc-00.txt*, Mar. 1998. Work in progress.

[16] Heinanen J., Baker F., Weiss W., Wroclawski J. "Assured Forwarding PHB Group". *RFC 2597*, Jun. 1999. Work in progress.

[17] Hou J., Tyan H.-Y., Wang B., Chen Y.-M. "QoS Extension to CBT". *9th International Workshop on Network and Operating Systems Support for Digital Audio and Video (NOSSDAV 99)*, Jun. 1999. (Also appared as an Internet Draft, draft-hou-cbt-qos-00.txt. Mar. 1999. Work in progress).

[18] Internet2 QoS Working Group Draft. "QBone Architecture (v1.0)". *http://www.internet2.edu/qos/wg/papers/qbArch/1.0/draft-i2-qbone-arch-1.0.html*, Aug. 1999. Work in progress.

[19] Jacobson V., Nichols K., Poduri K. "An Expedited Forwarding PHB". *RFC 2598*, Jun. 1999. Work in progress.

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

[21] Nichols K., Jacobson V., Zhang L. "A Two-bit Differentiated Services Architecture for the Internet". *Internet Draft, draft-nichols-diff-svc-arch-00.txt*, Nov. 1997. Work in progress.

[22] Quinn B. "IP Multicast Applications: Challenges and Solutions". *Internet Draft, draft-quinn-multicast-apps-00.txt*, Nov. 1998. Work in progress.

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

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

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

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

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

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

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

[30] 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.