A Functional Architecture for End-to-end Quality-of-Service in a Multi-domain Network*

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 235-99

Abstract

This report 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.

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

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 [24], RTP [17] or more recently QoSMIC [8], 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

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

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 report 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 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 report 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.

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) [4, 23, 19, 20, 21] and *differentiated services* (diff-serv) [3, 2, 14, 10]. 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 [14], 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

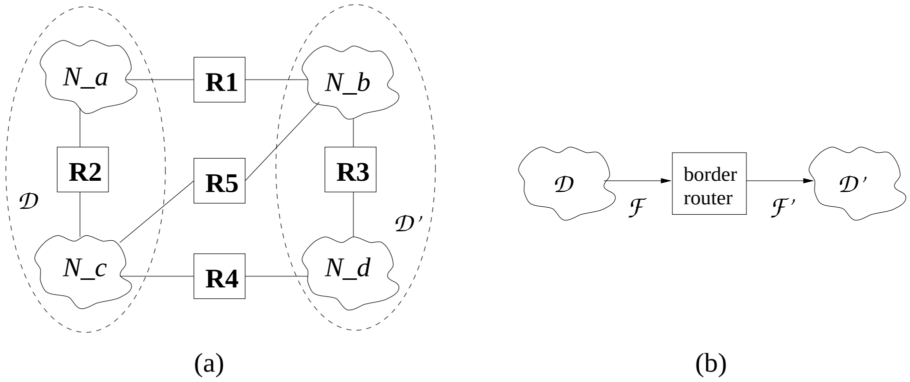

Figure 1: The system model.

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 [3], 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 [15, 13], while $\mathcal{D}'$ provides best-effort and *assured* services [11, 9]. The translation policy depends on the agreement between the adjacent domains.

A three layers architecture (Figure 2) describes how the routing is performed within the considered interconnected system [8]. 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 omogeneous 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

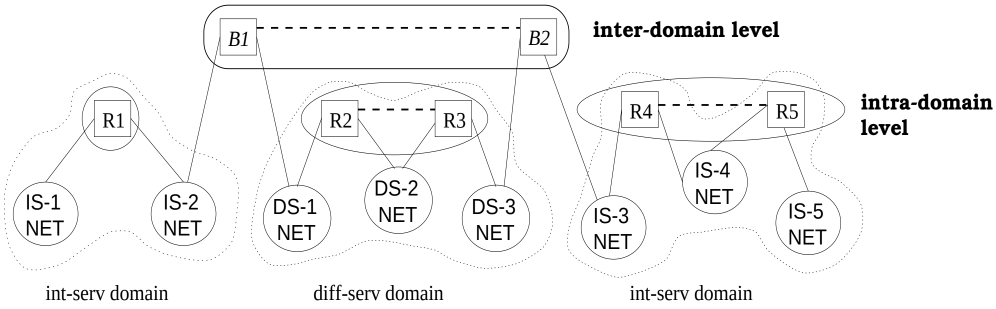

Figure 2: Routing architecture of QoS-capable routers.

[24], [4].

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 [22]. 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.

3.1.1 End-to-end layer

In this section, we outline the set of modules which directly operate on top of UDP protocol to provide multimedia application support and *lieson* service between host and network functions. To this purpose, we consider the following functions:

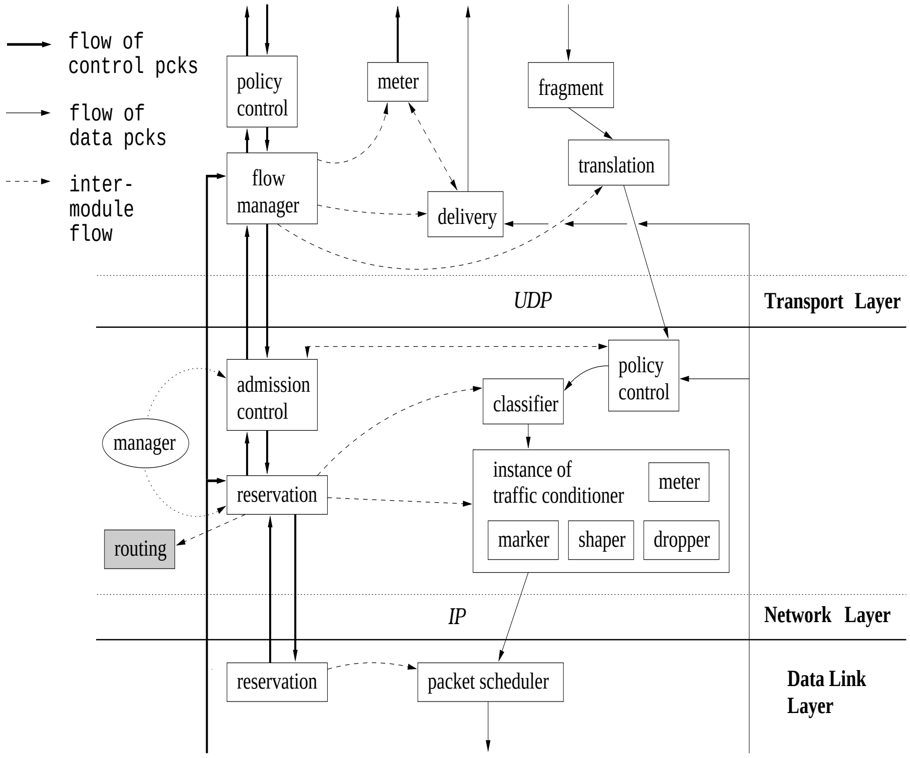

Figure 3: Functional architecture of a QoS-capable router.

policy control: the authorization of an application/user to access a QoS service is controlled. Information may be recorded for accounting purposes.

flow manager: 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` [20, 21] in the case of int-serv's, or a Per-Hop-Behavior (PHB) in the case of diff-serv's). This module is also in charge of the configuration of the `translation` and the `delivery` modules. Under specific QoS requirements, this module is responsible of performing end-to-end resource allocation. In fact, 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 [16]. The end-to-end parameters evaluation is performed in cooperation with the `meter` module.

meter: the transport layer can perform the end-to-end measurements of QoS parameters. 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 [17]. The constant measurement of QoS parameters can be used to perform (through the `flow manager`) the dynamic configuration of the `translation` module. Reports might be notified to the source application, to allow traffic profile adaptation according to the available network resources.

delivery: this module delivers the messages to the upper layer according to the specified QoS requirements. When dealing with a multimedia flow, this module is in charge of keeping all the media streams synchronized.

fragmentation: this module fragments the application messages into transfer units which possibly match the size of network packets. 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 [3]; otherwise, the fragmentation at this layer generates fragments as large as the smallest fragment of the traversed networks [24].

translator: this module marks the data packets in order to be processed according to the specified requirements. The `translator` module is also in charge of changing the data encoding (audio/video encoding) to meet the underlying available resources or mapping multicast addressing onto unicast addressing scheme [17].

The current version of RTP [17] covers the functions of the modules `flow manager`, `delivery` and `meter`. If RTSP [18] is used on top of RTP, the modules `delivery` and `flow manager` must be partially duplicated to manage the multimedia flows as aggregate of media streams.

3.1.2 The network layer

In this section, we consider the set of modules which operate on top of the IP protocol and transform a best effort router into a QoS-*capable* router. To this purpose, we can locate at the network layer the following functions:

admission control: for all the incoming requests (from both the network and the upper layer), this module checks, by contacting the `policy control` module, the validity of the request, verifies whether or not the locally available resources are sufficient to satisfy the request and delegates the `resource reservation` module to perform the reservation. `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 [24].

resource reservation: this module performs resource reservation whenever a request for a new QoS flow or behaviour aggregate is received. By interpreting either the `TSpec`, in case of IS requests, or the traffic profile and class specified in the *Reservation Request Object* (RRO) packets [12], in case of DS requests, this module reserves buffer space and generates control information to configure the underlying `packet scheduler`. The `resource reservation` module configures the `classifier` and the `traffic conditioner` modules, in order to ensure the proper processing of the data packets during the data transfer phase.

policy control: this module is activated whenever a new data packet is received, either from an adjacent router or from the upper layer. It checks the correctness of a specified portion of the packet header and its coherence with the QoS requested at the set-up time.

manager: this module allows to perform the preliminary configuration of the `admission control` and `resource reservation` modules. 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.

packet classifier: this module selects the data packets in a traffic stream 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 by the `resource reservation`. The `classifier` directs the packets matching the specified rules towards the proper instance of the `traffic conditioner`.

traffic conditioner: one copy of the `traffic conditioner` is instantiated for each class of service (in the case of diff-serv) or for each data flow (in the case of int-serv). The module's structure we are considering derives from the diff-serv proposal, [3], and is composed of four sub-modules: `marker`, `shaper`, `dropper` and `meter`. When acting as a component of a border router, the `traffic conditioner` remarks all packets crossing a domain boundary. This task is performed by the `marker` sub-module, which is also responsible of the marking of all the out-of-profile packets which can, for instance, be demoted to a worse service class. The re-marking policy may be either described in the specification of the service, or agreed among the administrators of the adjacent network domains. The `meter` sub-module measures the characteristics of the packet streams which belong to a given traffic class, and compares them with the traffic profile of that class. The measurements may be used to dynamically adapt the resource reservation and the packet scheduling policies. The `shaper` sub-module re-shapes traffic bursts according to the traffic profile of the service class to which packets belong. The `dropper` sub-module intervenes by discarding the packets whenever the `shaper` and the `marker` have failed, respectively, in re-shaping or demoting the incoming flow.

The protocol RSVP [24] partially covers the functions of the `reservation` module. The Bandwidth Broker described in [15] covers the `reservation` and the `admission control` modules.

Figure 3 shows a complex architecture of the network layer. Complexity is bringing together a high computational cost which is mainly concentrated in the modules `reservation`, `classifier` and `policy control`. 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.

3.1.3 The data link layer

At the data link layer the following modules are located:

resource reservation: according to the requests generated by the corresponding module at the network layer, this module reserves buffer space and configures the `packet scheduler` to implement the appropriate forwarding policy for the different classes of traffic.

packet scheduler: this module decides the order in which packets are forwarded. Decision is made according to the class of traffic to which packets belong. Packets can be ordered in the output queues according to a FIFO, priority or weighted fair policy [13].

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-

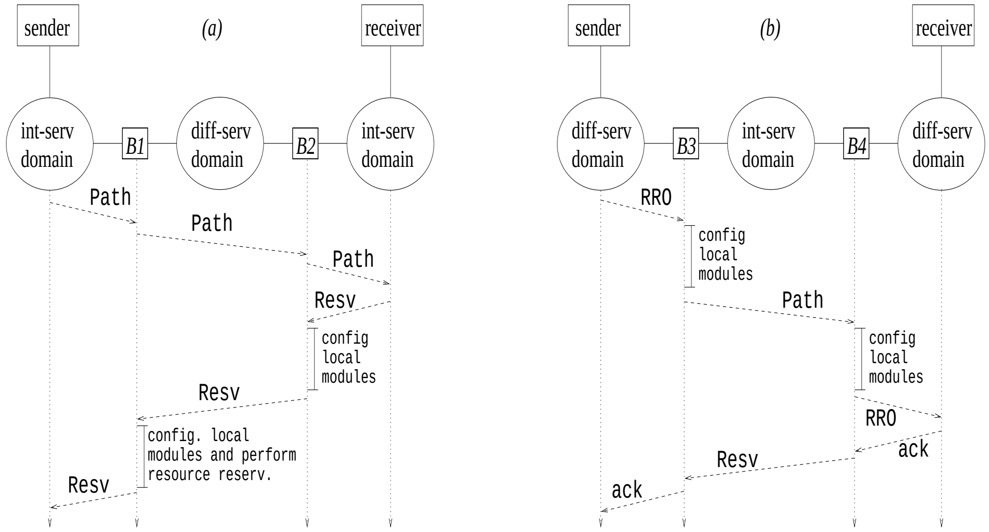

Figure 4: Multi-domain configurations and resource reservation procedures.

serv domain means that the ISP a-priori decides the amount of resources to reserve for each PHB by running the resource reservation setup procedure [12]. 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.

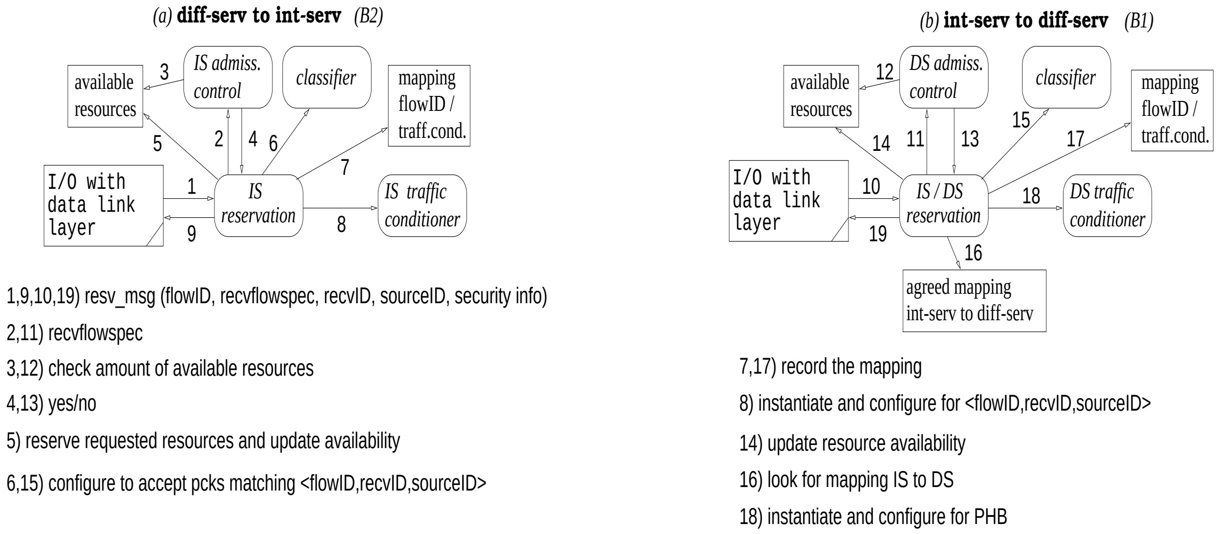

Figure 5: Border router control flows for the configuration IS-DS-IS.

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* [3]. 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 `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

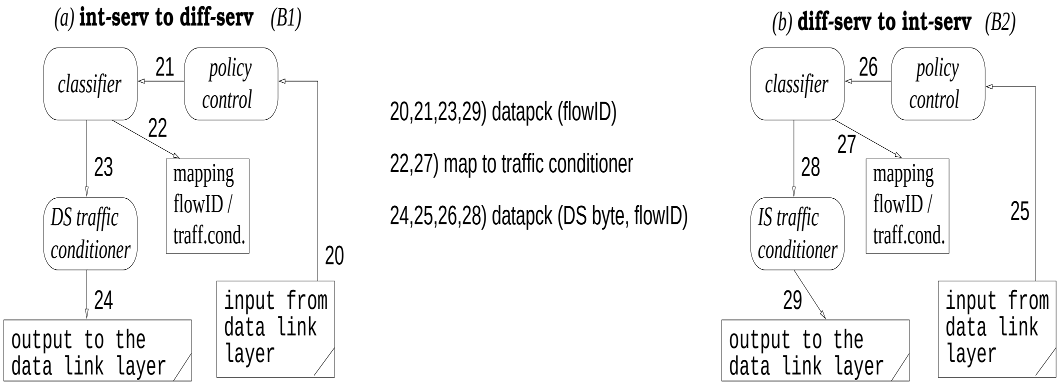

Figure 6: Border router data flows for the configuration IS-DS-IS.

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).

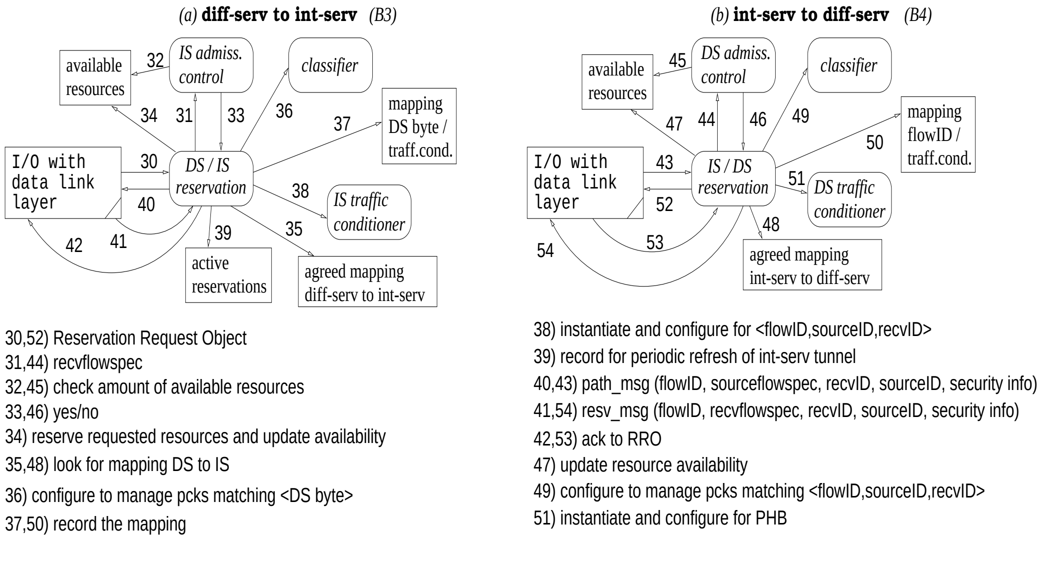

Figure 7: Border router control flows for the configuration DS-IS-DS.

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 [12] 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 [5], 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 [7] and CBT [1]). So far, very few proposals have been presented to provide multicast QoS (e.g. RSVP [24], RTP [17], QoSMIC [8]). 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. In Figure 8, we show the functional architecture

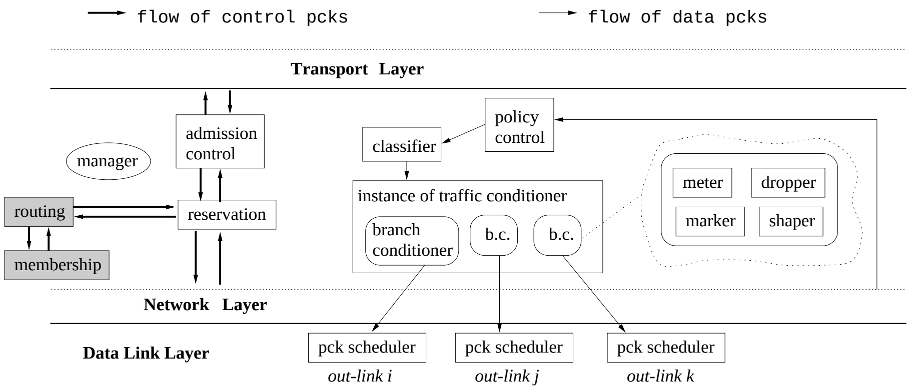

Figure 8: QoS multicast functional architecture.

of the network layer of a generic QoS capable router which supports multicast traffic. The other layers of the protocol stack are slightly changed. 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.

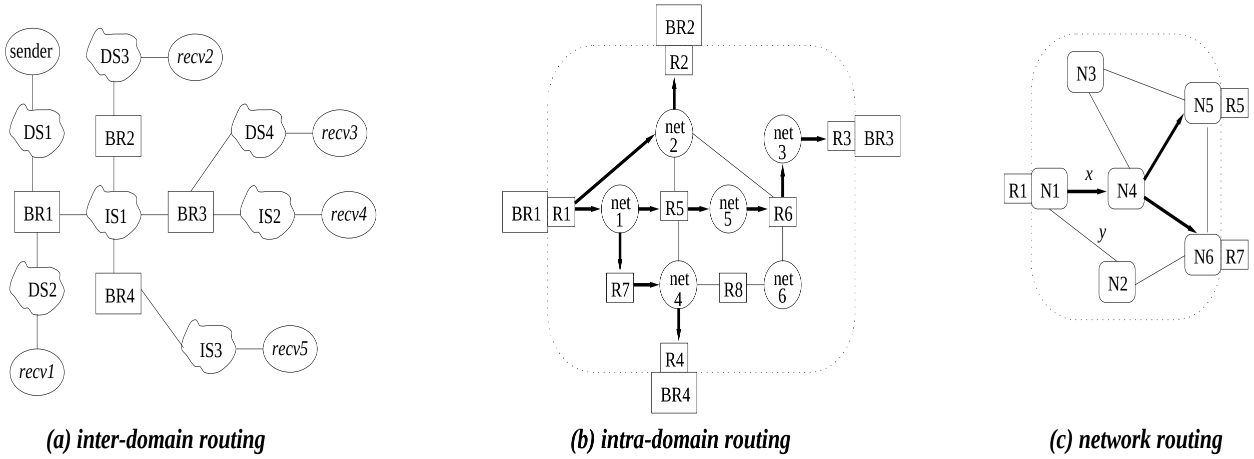

Figure 9: Example of multi-domain routing.

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.

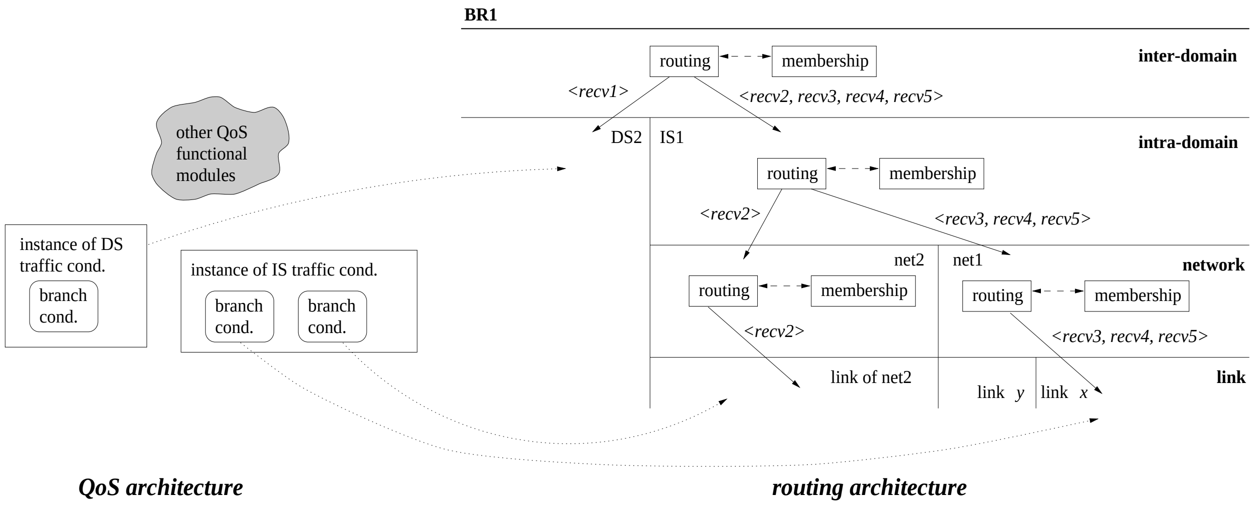

Figure 10: The QoS and routing architectures of BR1.

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.

It should be noted that IGMP [6] 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.

5 Concluding remarks

This report 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 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 QoS-capable routers will be probably designed and deployed.

References

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

[2] 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.

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

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

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

[6] 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.

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

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

[9] 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.

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

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

[12] 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.

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

[14] 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.

[15] 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.

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

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

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

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

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

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

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

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

[24] 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.