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

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

[1] Dip. di Elettronica e Automatica, Università degli Studi di Ancona
cangio@popcsi.unian.it
[2] Dip. di Scienze dell'Informazione, Università degli Studi di Milano
{pagani, rossi}@dsi.unimi.it

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

1 Introduction

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

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

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

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

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

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

2 The System Model

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

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

3 The Reference Architecture

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

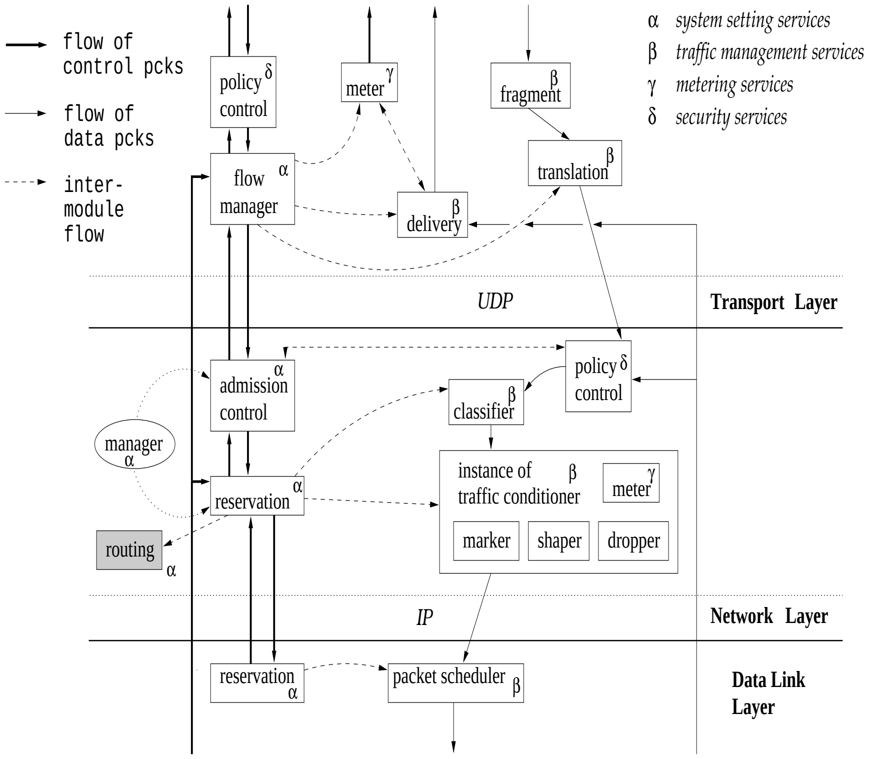

Figure 1. Functional architecture of a QoS-capable router

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

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

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

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

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

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

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

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

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

3.1 Supporting QoS-Multicast Traffic

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

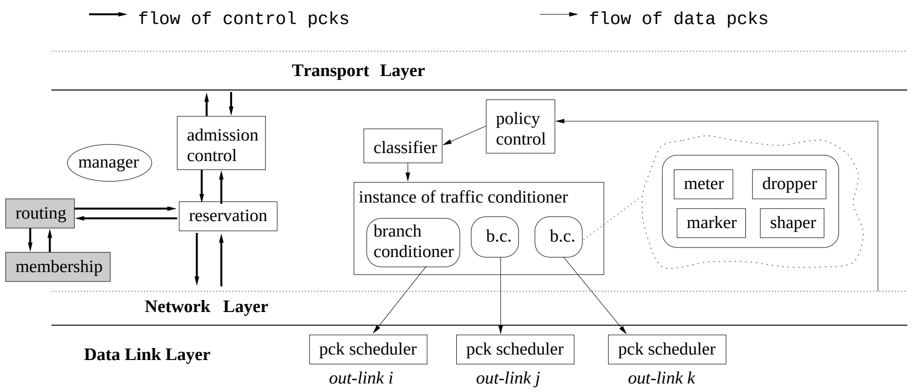

Figure 2. QoS multicast functional architecture

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

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

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

4 The Architecture Implementation

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

4.1 The Diff-serv Approach

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

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

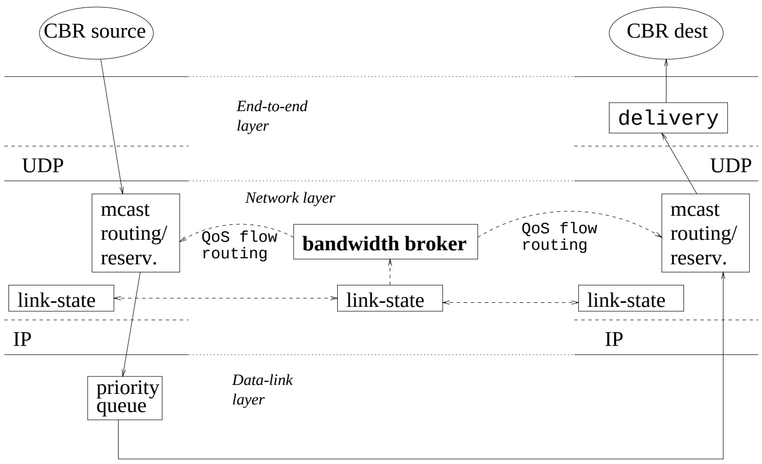

Figure 3. Layout of the architecture implementation for the DS model

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

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

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

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

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

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

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

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

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

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

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

4.2 The Int-serv Approach

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

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

As the packet scheduling policies we currently consider:

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

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

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

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

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

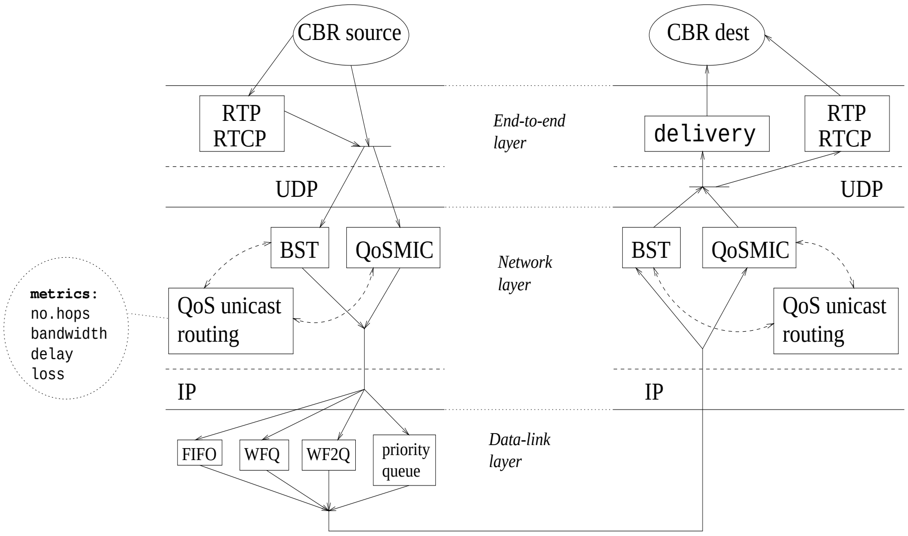

Figure 4. Layout of the implemented architecture

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

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

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

5 Performance Evaluation

5.1 The Diff-serv Approach

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

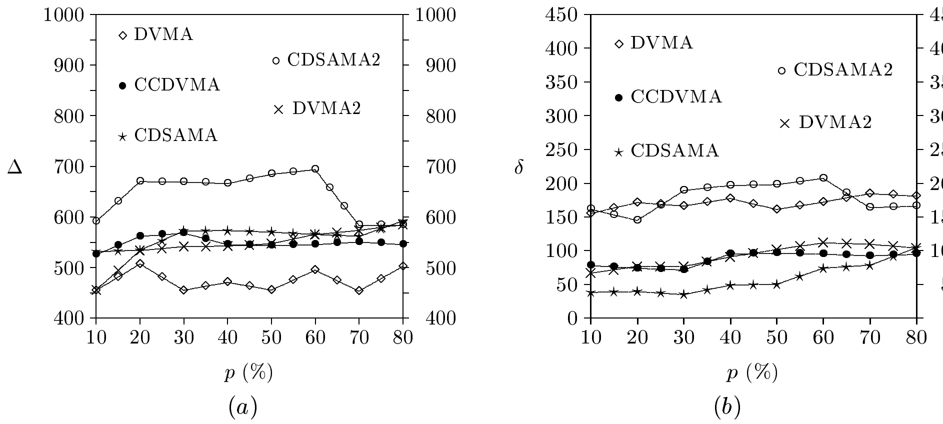

Figure 5. (a) Maximum delay towards 2 destinations in a 20-nodes network. (b) Maximum fair delay towards 2 destinations in a 20-nodes network

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

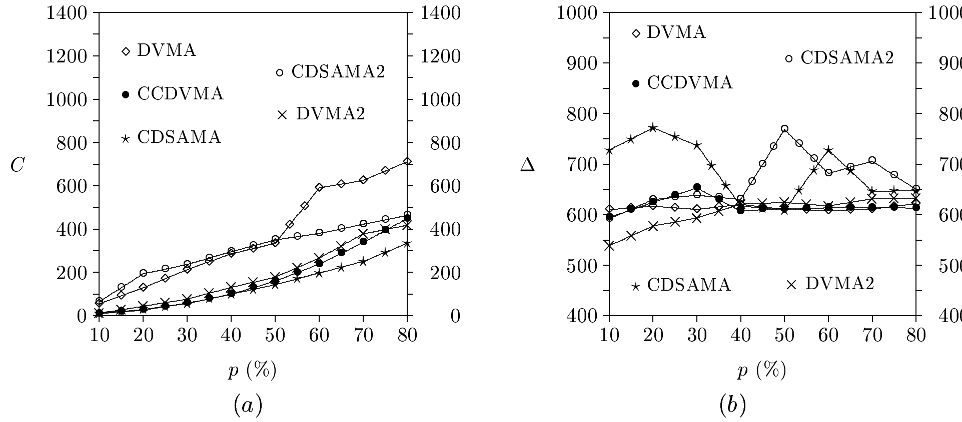

Figure 6. (*a*) Cost of the tree connecting 2 destinations in a 20-nodes network. (*b*) Maximum delay towards 4 destinations in a 20-nodes network

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

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

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

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

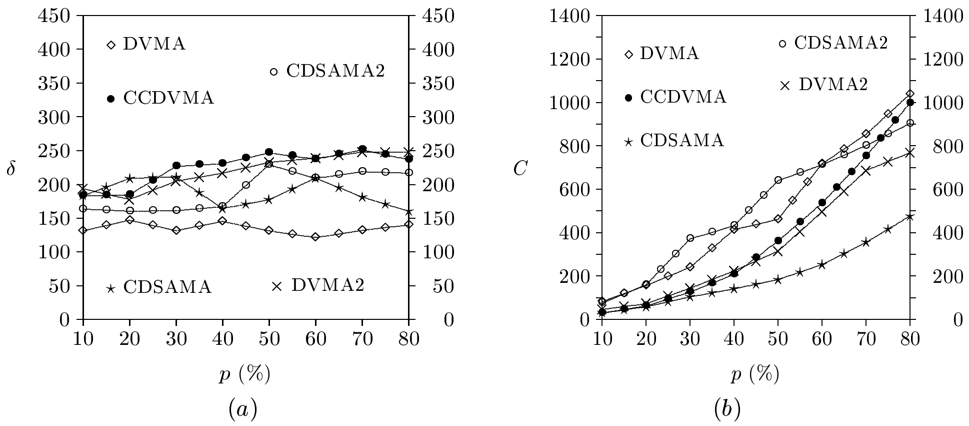

Figure 7. (*a*) Maximum fair delay towards 4 destinations in a 20-nodes network. (*b*) Cost of the tree connecting 4 destinations in a 20-nodes network

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

5.2 The Int-serv Approach

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

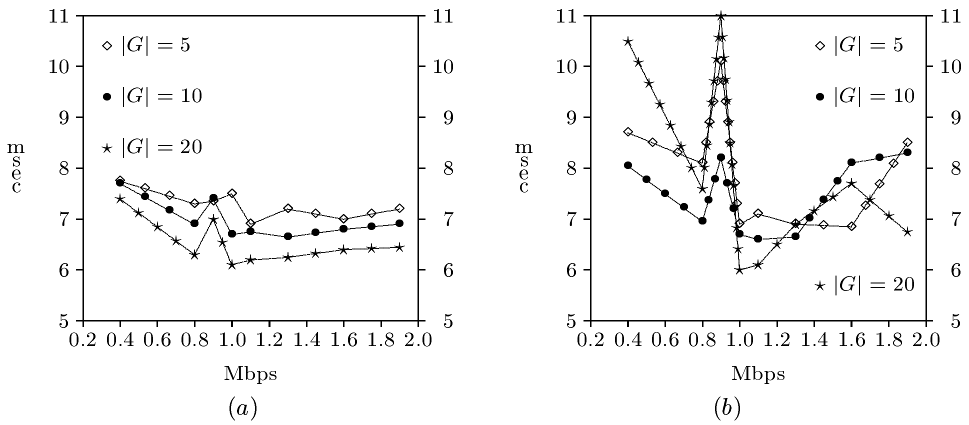

Figure 8. (*a*) Average end-to-end delay with PQ. (*b*) Average end-to-end delay with WF2Q

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

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

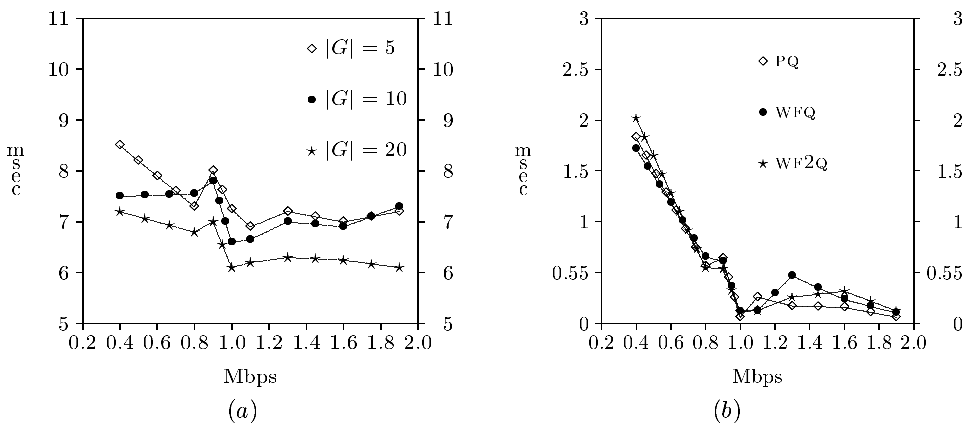

Figure 9. (a) Average end-to-end delay with WFQ. (b) Jitter for $|G| = 10$ and different packet scheduling policies

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

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

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

6 Concluding Remarks

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

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

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

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

References

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