Implementation Schemes for Multicast Bandwidth Brokers in Multidomain Networks [1,2]

Andrea Borella [a], Giovanni Cancellieri [a], Elena Pagani [b], Gian Paolo Rossi [b]

[a] *Dip. di Elettronica e Automatica, Università degli Studi di Ancona*

[b] *Dip. di Scienze dell'Informazione, Università degli Studi di Milano*

Abstract

This paper describes two new approaches that are capable of ensuring delivery guarantees to multicast sessions on IP-based networks, according to the diff-serv model. We present the CDSAMA and DVMA2 QoS-sensitive multicast routing algorithms that may be used to implement a centralized bandwidth broker and support multiple QoS constraints, namely, bandwidth, delay and fair delay. We describe an end-to-end *Call Admission Multicast Protocol* (CAMP) that operates as a distributed bandwidth broker and adopts an approach which is simple, scalable, operates on a per-call basis and supports the group membership dynamics. We discuss some performance evaluations obtained by implementing the proposed solutions in a simulation environment.

Key words: Quality-of-Service; multicast; admission control; differentiated services; bandwidth broker

1 Introduction

A great deal of efforts has been recently devoted to provide through the Internet a transport platform capable of satisfying the Quality-of-Service (*QoS*) requirements of the emerging multimedia and real-time applications. This goal

[1] This work was supported by the MURST under Contract no.MM09265173 "Techniques for end-to-end Quality-of-Service control in multi-domain IP networks".

[2] A preliminary version of this paper was published in the Proc. International Workshop on QoS in Multiservice IP Networks (9).

cannot be achieved without the introduction of some traffic differentiation policy, that changes the best-effort nature of the Internet protocols into a new generation of protocols providing QoS guarantees (e.g., in terms of delay, bandwidth, jitter, fair delay (13; 1)), and that addresses the following general application requirements:

multicast: most of the video, voice and multimedia applications require the support of one-to-many or many-to-many interaction schemes (12);

scalability: the QoS-capable transport mechanism should scale to the entire Internet and to large multicast groups;

dynamic sessions: the QoS-capable transport mechanism should accept the dynamic opening/closing of QoS sessions on an on-demand basis;

group membership: the membership of the multicast group may be *static*, i.e. the result of an announcement and subscription process through, for instance, `sdr` (20), or *dynamic*. In the former case, the source can be aware of the group membership, while, in the latter, the source is generally unaware of both the membership and the cardinality of the group over time.

The Int-Serv (IS for short (11)) and Diff-Serv (DS for short (5)) differentiation approaches, which have been proposed by the IETF, do not completely satisfy the above requirements and, although they consider multicast communications, in practice, they do not offer a satisfactory solution for managing group membership and session dynamics. IS can dynamically provide QoS guarantees on a per-flow basis through RSVP (35). To do that, it requires substantial changes in the router architecture; each router maintains per-flow status information by running a hop-by-hop information exchange protocol. This makes RSVP hard to scale and unusable, in practice, when large multicast groups are considered. DS, on the contrary, considers flow aggregates and all packets belonging to the same aggregate receive the same network service on the base of the TOS byte in their IP header (DS byte in IPv6) (24), that identifies, and differentiates, the packet's *Per-Hop-Behaviour* (PHB). The DS model is simple and more scalable because it confines the QoS complexity at the *boundary routers* (i.e., the routers of the access network or the routers on a domain border) and leaves the core routers nearly unchanged. This approach requires that an end-to-end admission mechanism (the *bandwidth broker* (25)) dynamically controls and limits the amount of incoming flows to be assigned to a given aggregate. So far, such an end-to-end admission mechanism has not been designed for multicast-enabled sessions. As a consequence, the current QoS multicast experiences (e.g., QBone (21)) still consider a static environment where the amount of multicast QoS sessions and the group membership are both statically defined in a sort of permanent multi-point virtual circuit. Moreover, inter- and intra-domain communications are not considered in details.

This paper describes new approaches that are capable of ensuring delivery guarantees to multicast sessions on IP-based networks and satisfy the above requirements. We present the CDSAMA and DVMA2 QoS-sensitive multicast routing algorithms that may be used to implement a centralized bandwidth broker and support multiple QoS constraints, namely, bandwidth, delay and fair delay. We describe an end-to-end *Call Admission Multicast Protocol* (CAMP) that operates as a distributed bandwidth broker and adopts an approach which is simple and scalable, that operates on a per-call basis and supports the group membership dynamics. We discuss some performance evaluations obtained by implementing the proposed solutions in a simulation environment. As far as we know, this is the first attempt to address all the above requirements for multicast traffic in DS networks. The simulation results are encouraging and indicate that the approaches are efficient for CBR traffic.

The paper is organized as follows: in Section 2, we introduce the system model considered throughout the work and the adopted notation. In Section 3, we present the CDSAMA and DVMA2 multicast routing algorithms, and we discuss the results of their performance measurements. In Section 4, we describe the CAMP protocol and the experimental results achieved by implementing it in the NS-2 simulation framework. Section 5 concludes the work.

2 System model

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 proposed: *integrated services* (int-serv, IS) (11) and *differentiated services* (diff-serv, DS) (5). 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 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 a consequence of a classification and, possibly, conditioning process. Each packet, through the use of the TOS/*DS* byte field of the IP header (24), 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 (PHB); they constitute a *behaviour aggregate*.

Packets are forwarded through an interconnected system (Figure 1(a)), composed of a set of networks connected by routers. Routers allow the forwarding of messages through different networks and are assumed to implement the

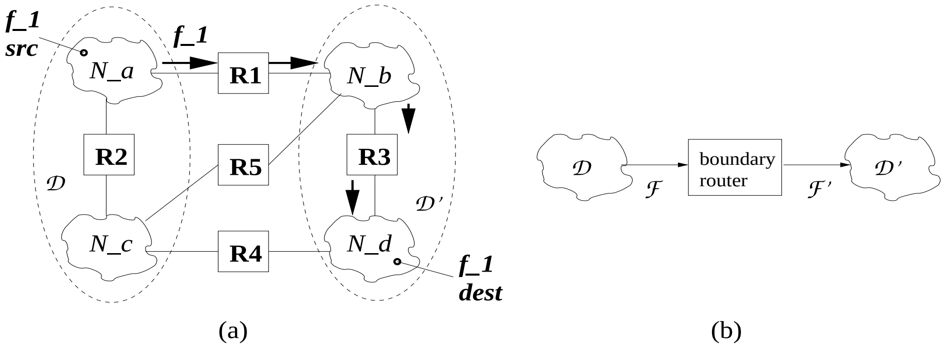


Figure 1. The system model

QoS services. According to the notation adopted for diff-serv (5), 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), R_1 is the ingress router for the flow f_1 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. We denote as *boundary* routers (BRs) those routers connecting contiguous network domains. In order to guarantee the requested end-to-end QoS (Figure 1(b)), the *boundary* routers (R_1, R_4 and R_5 between the domains D and D' in Figure 1(a)) are responsible for mapping the service provisioning and forwarding policy F to which a flow is subject in one domain into the model and policy F' which have to be applied in the adjacent domain. The translation policy depends on the agreement between the adjacent domains. In this work we assume that hierarchical routing is adopted to support inter-domain communications, that is, the routing and membership functionalities are replicated at the inter-domain, the intra-domain and the network levels (14; 18; 2).

In (25) an architecture is proposed to provide QoS according to the diff-serv model, that exploits so called *bandwidth broker* (BB) agents to perform the functionalities of call admission control, resource reservation and system configuration. A BB exists for each domain. According to the reference architecture described in (26), the BBs span the end-to-end and network layers in the OSI model. In Figure 2, we show the functional architecture that we consider in this work, and we detail the BB internal structure, in accordance with the proposal presented in (34). The BB provides the applications with the interface to access the QoS services. It may exploit the services of both a membership protocol, and the routing protocols to acquire knowledge about network topology and resource availability. It performs the configuration of the intra-domain routers for what concerns the routing and packet scheduling functions. When the QoS aggregate spans multiple domains, an inter-domain protocol is executed amongst peer BBs, to guarantee the proper configuration

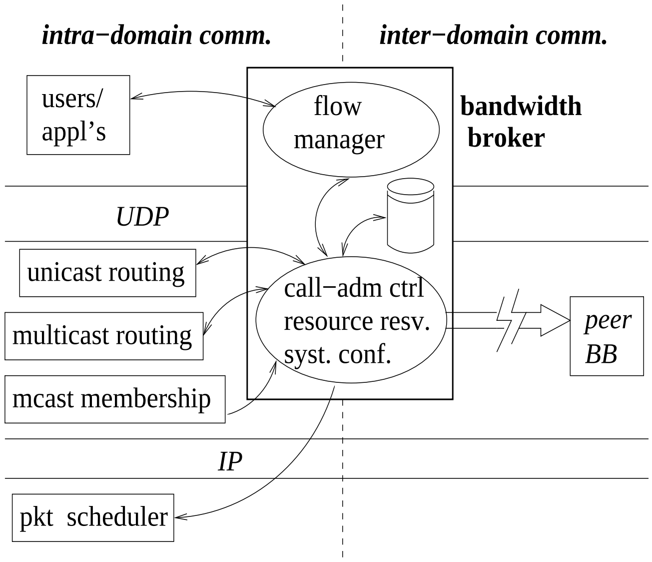


Figure 2. Framework architecture

of the transit and destination domains.

We assume that a session announcement protocol (e.g., `sdr` (20)) is used to announce the needed session information, such as the transmission start time and the multicast address of the destination group. Receivers perform an explicit join procedure so that the transmission source is aware of the membership of the group of recipients before the transmission starts. All the recipients receive the same set of microflows. In this paper, we consider sources generating CBR traffic. VBR traffic can be supported by considering it equal to a CBR traffic profile with rate set to the VBR peak rate. This is the easiest way to deal with VBR traffic; this approach is also adopted in (3; 4).

2.1 Related works

So far, very few proposals have been presented about the BB implementation. The QBone testbed (21), under development by the Internet2 QoS Working Group, involves a BB that is initially "implemented" by a human operator who manually configures the system. Scenarios have been figured out to design an inter-BB communication protocol (SIBBS) (34). Anyway, no technique is proposed to automatically perform intra-domain configuration. In this paper, we discuss two mechanisms to implement the resource reservation, call-admission control and system configuration functionalities both at intra-domain and inter-domain level, which adopt either a centralized or a distributed approach.

Under the centralized approach we propose QoS-sensitive multicast routing algorithms that consider multiple constraints (namely: bandwidth, delay and fair delay constraints) to characterize a tree structure. Algorithms similar to those we propose are the DVMA algorithm (29) and the CCDVMA algorithm (19). DVMA (Delay Variation Multicast Algorithm) 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 that complies with the former constraint at least. Dijkstra's algorithm is encapsulated in DVMA to perform the first attempt tree search minimizing the source-destination delays. If the fair delay of that tree is greater than the bound, DVMA operates by selecting some alternative branches so that the resulting tree complies with both constraints. CCDVMA (Cost Conscious Delay and Delay Variation Multicast Algorithm) has been developed to include bandwidth within the goals of QoS multicast routing. CCDVMA is similar to DVMA. In fact, CCDVMA as well uses the Dijkstra's algorithm to identify a first attempt multicast tree. But in this case, Dijkstra's tree is computed considering the link bandwidth availability instead of the delays. After that, CCDVMA selects a given number of possible alternative routes, to join the nodes not reached by the tree, trying to satisfy the delay and fair delay constraints. 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 algorithms proposed in section 3 differentiate from DVMA since they perform the tree selection taking into account both QoS and efficiency requirements, whereas the latter is not constrained by the cost parameter. They also try to extend and improve some basic characteristics of CCDVMA, increasing the capability of finding alternative routes, by fairly weighting the path search criteria that each of the above mentioned constraints would impose if applied on its own.

Under the distributed approach, we propose a signaling technique that tests for resource availability without requiring state information maintenance in the routers. A similar approach is adopted by the PCP protocol (10) and applied to point-to-point video/voice sessions. Anyway, PCP can only be used to provide bandwidth reservation in the case of unicast communications, while it does not involve mechanisms to deal with multicast issues, differently from the protocol proposed in section 4.

Preliminary versions of both the centralized and the distributed algorithms have been proposed in (27; 9). Here, we explain the algorithms with more detail, we discuss their suitability in the framework of the diff-serv model, and we present a larger set of simulation results.

3 Centralized Bandwidth Broker

We have devised three algorithms that can be embedded into the bandwidth broker. They build a *multicast routing tree* structure along which the behaviour aggregates are forwarded, taking into account the QoS requirements. The algorithms have a centralized control. In this section, we initially describe the layout of the BB functioning. We specify the BB interfaces according to the architecture shown in Figure 2. We discuss the intra-domain and the inter-domain configuration process [3], under the assumption of static group membership which is known to the flow source before the system configuration is performed. We are currently extending the algorithms to support dynamic membership changes. Finally, we describe in detail the algorithms for the QoS-sensitive multicast routing.

An application requiring QoS contacts its domain BB (Figure 3) by specifying the profile of the traffic it is going to generate, the list of subscribing receivers and the needed QoS constraints. The algorithms that we propose in

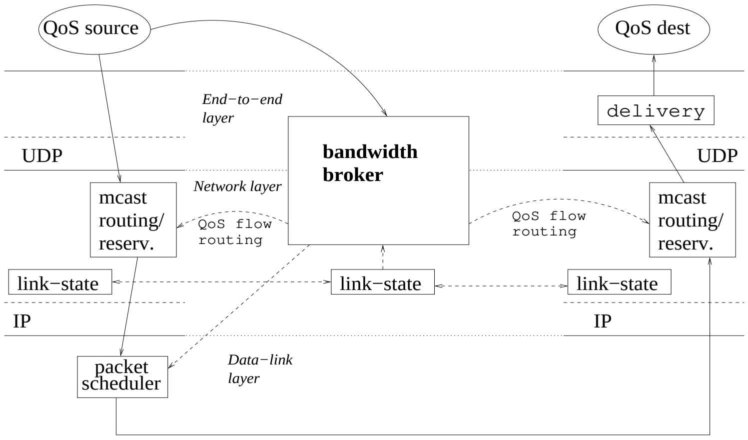


Figure 3. Layout of the architecture implementation for the centralized BB

the following consider three constraints, namely, the bandwidth availability, the end-to-end delay and the fair delay. The bandwidth availability is represented by the link *cost*, which is the ratio between the reserved bandwidth and the total link capacity. The fair delay is the maximum difference between the end-to-end delays perceived by two different receivers. The algorithms only support *homogeneous* recipients, having the same QoS requirements.

The BB needs the complete knowledge of the topology at both the intra-domain and the network levels, and of the statical QoS metrics. To this purpose, either a link-state routing protocol can be exploited, or an intra-domain

[3] Anyway, we do not examine in detail the intra-domain and inter-domain communication protocols, as they are out of the scope of this paper.

protocol must be executed such that the routers in the domain send their routing tables to the BB. By using that information, the proposed algorithms produce tree structures which satisfy the specified constraints.

The BB performs the intra-domain configuration by overwriting the routing tables of the in-tree routers, so that they carry out the packet forwarding along the characterized tree. Moreover, the BB configures the packet schedulers of the in-tree routers so that they provide the appropriate forwarding policy.

To perform the inter-domain system configuration and guarantee the approach scalability we exploit the hierarchical routing scheme. As the algorithms have a centralized control, a *master* BB (BB_M) has to be designated among the BBs of the source and destination domains. BB_M must know the system topology at the inter-domain level, together with the estimates of the available bandwidth and the end-to-end delay on the path that connects two BRs that are one-hop away through a domain. Moreover, BB_M knows the BBs responsible for the configuration of the different domains. When BB_M receives a service request forwarded to it from the BB of the source's domain, it computes the multicast tree at the inter-domain level and it configures the in-tree boundary routers accordingly. BB_M also contacts the BBs responsible for the transit and the destination domains. Those BBs, in turn, compute the appropriate paths at the intra-domain level and perform the domain configuration as described above. The inter-domain protocol adopted among peer BBs could be similar to the SIBBS protocol (34).

In the following, we define the maximum source-destination delay Δ_{max} as the age of obsolescence of the packet information, which is reached at time Δ_{max} after the transmission of the packet itself. Moreover, 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 pair of destinations belonging to the same multicast group. Inside a network modeled by the graph $G = \langle V, E \rangle$, 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 corresponding average delay $D(r)$, the constraints considered by the algorithms we propose can be analytically written as follows:

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

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

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

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. In this context, they offer a possible way of obtaining reasonably good performance considering either the service provider necessities (which needs to increase the resource utilization at the most) or the user point of view (that asks for high QoS levels).

3.1 CDSAMA algorithms

The goal of *CDSAMA (Cost Delay Sensitive Applications - Multicast Algorithm)* (7) is the (heuristic) minimization of the cost of a multicast tree which satisfies both maximum delay and fair delay constraints. Its inputs are represented by the source, the multicast group, the constraints (1), (2), (3) and two integers k and h whose utilization will be explained in the sequel. CDSAMA encapsulates the Dijkstra's algorithm, that is utilized to define the first attempt tree. It is well known that Dijkstra is a shortest path routing, providing optimized solutions. However, in principle, it can be applied in different metric contexts to minimize the target parameter for every source-destination connection in a multipoint routing path. Consequently, within the CDSAMA family, two alternative approaches have been considered. In the first case we use Dijkstra to minimize the cost of every source-destination connection, so that the target parameter is the link cost. CDSAMA-C version of the algorithm adopts this policy. In a different case, Dijkstra minimizes the delay on every source-destination path. This technique is implemented in CDSAMA-D. Both versions, after Dijkstra, operate to verify the availability of alternative paths, reaching the same multicast group, complying with QoS requirements. Those paths are obtained by deforming the starting tree, using a heuristic strategy that is illustrated here. For the sake of brevity, only CDSAMA-C is described in detail, CDSAMA-D being developed analogously.

STEP 1: Being s the multicast source and M the multicast group in $G\langle V, E\rangle$ let S (with starting value $S = [0]$) be the vector containing the list of branches of the multicast tree and U (with starting value $U = [s]$) the vector including the nodes of G belonging to the same tree.

STEP 2: Set $T = T_0$, where T_0 is the multicast tree calculated by Dijkstra's algorithm in G, operating on the link costs. Let q be the destination node characterized by the highest source-destination cost in T_0 among those characterized by a source-destination delay lower than, or equal to, Δ_{max}. If such a node does not exist, the algorithm selects, as node q, the one having the lowest delay Δ, independently of its corresponding cost.

STEP 3: Identify k paths $(P_0, P_1, ..., P_{k-1})$, from s to q, having minimum cost and satisfying constraint (1). The first path P_0 is that obtained by Dijkstra in G, whereas every other path is calculated by applying the same algorithm in the G' graph, that is a subset of graph G where a link belonging to P_0 has been selectively erased. Thus, for different links of P_0 we have different graphs G' and, therefore, different alternative paths. In particular, P_1 is obtained by running Dijkstra on a subset of G where the first link of P_0 is cancelled, P_2 is calculated on a different subset of G, where the second link of P_0 is not present and so on, until $k-1$ alternative paths are identified.

STEP 4: Repeat the following steps, from 5 to 13, for every path P_i with $0 \leq i \leq k-1$. In those steps a multicast tree T_i is built for every path P_i identified at step 3. T_i includes all the elements of M.

STEP 5: Initialize $T_i = (U_i, S_i)$ that is the temporary tree, initially composed by path P_i only, where U_i and S_i are respectively the set of nodes and the set of edges belonging to P_i.

STEP 6: Let $M = M - (M \cap U_i)$ be the set of destinations to be reached by T_i (not already connected). Until $M \neq \emptyset$ repeat steps from 7 to 13.

STEP 7: Select node $u \in M$. From steps 8 to 13 it will be reached by T_i. For every node $v \in U_i$, repeat steps 8-10.

STEP 8: Build a new graph G'', by erasing in G all the nodes belonging to $U_i - \{v\}$ and all the links belonging to S_i.

STEP 9: Identify the first h minimum cost paths connecting v and u in G''. Those paths are calculated using the same procedure used at step 3.

STEP 10: Among the h paths calculated at step 9 select the one characterized by the minimum corresponding Δ delay; if more than one path complies with this condition, select the path implying the minimum δ; if more than one path satisfies both the previous conditions, select the one having the minimum cost. Label the selected path by Q_v.

STEP 11: Among all the Q_v paths obtained with the previous cycles, select the best, according to the same criteria indicated at step 10, and label it as path Q.

STEP 12: Update the temporary tree $T_i = (U_i, S_i)$ by including in it all the nodes and the edges belonging to Q. In this way, the path P_i connecting s and q, by incorporating path Q, reaches destination u. In particular, in U_i and S_i are inserted all the nodes and links of Q respectively.

STEP 13: Update M so that, being $M = M - \{u\}$, it contains the destination to be reached by the tree T_i only. If $M \neq \emptyset$ then go to step 7, if $M = \emptyset$ but $i \neq k-1$ then go to step 5, otherwise go to step 14.

STEP 14: Let T the best tree between the trees T_i, with $0 \leq i \leq k-1$, according to the criteria adopted at step 10.

STEP 15: Output T and stop.

In practice, when possible, the algorithm identifies a tree that is compatible with the constraints on the maximum delay and maximum fair delay. Among all the trees that satisfy such limits, it chooses the cheapest. When the algorithm fails, in any case, it suggests a "best effort" solution, possibly close to the optimal one.

A possible evolution of DVMA is represented by the *DVMA2* algorithm (8). It is, once again, a 3-constrained multicast algorithm, which includes an efficient tree search strategy, 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 is reduced. The output of the algorithm is represented by a tree minimizing the fair delay, even when the relative constraint (2) is violated. On the basis of the same assumptions and formalisms adopted in section 3.1, the algorithm works as illustrated in the sequel.

STEP 1: Let s be the multicast source and M the multicast group within the $G\langle V, E\rangle$ graph. Let S (with starting value $S = [0]$) the vector containing the list of branches of the multicast tree and U (with starting value $U = [s]$) the vector including the nodes of G belonging to the same tree.
STEP 2: Set $T = T_0$, where T_0 is the multicast tree calculated by Dijkstra's algorithm in G, operating on the link delays; let w be the node characterized by the highest source-destination delay in T_0.
STEP 3: Calculate the first k shortest paths (in terms of delay) from s to w, so that all of them comply with constraint (1); those paths are labeled by $P_0, ..., P_{k-1}$. The first path P_0 is the one found by Dijkstra in G, whereas the others are identified by applying Dijkstra again on graph G'. The latter is obtained from G by excluding, one at a time, the links belonging to path P_0 from s to w.
STEP 4: For every path P_i, with $0 \leq i \leq k-1$, set a corresponding δ_i. The latter equals 0 if no other destinations are reached by P_i, whereas it equals the maximum difference of the source-destination delay experienced by all the nodes in P_i belonging to M. The k paths P_i are then grouped in two sets YES and NO, according to the following rule: $P_i \in YES$ if $\delta_i \leq \delta_{max}$, otherwise $P_i \in NO$.
STEP 5: For every path $P_i \in YES$ repeat steps from 6 to 14. The goal consists in the construction of a complete tree T_i for every path P_i containing all the elements of M.
STEP 6: Let $T_i = (U_i, S_i)$ be the temporary starting tree, where U_i and S_i contain respectively the nodes and the links of P_i.
STEP 7: Update $M = M - (M \cap U_i)$, so that M includes the destinations waiting to be connected to T_i. Until $M \neq \emptyset$, repeat steps 8-14.
STEP 8: For every couple of nodes $u \in M$ and $v \in U_i$, repeat steps 9-11.
STEP 9: Build a new graph G'', obtained from G by erasing all the nodes belonging to $U_i - \{v\}$ and all the links belonging to S_i.
STEP 10: Identify the first h paths from v to u in G'', following the same

criteria adopted in step 3.

STEP 11: Among all the h paths calculated in step 10, whose connection to the multicast tree does not imply any violation of constraints (1) and (2), the route having the minimum cost is chosen. In case of multiple paths characterized by the same minimum cost, the route having the minimum δ is selected. Otherwise, if all the available paths imply a resulting δ greater than δ_{max}, choose the route with the minimum δ and, in case of equivalent paths, the one with the minimum cost. The resulting path is labeled by Q_v.

STEP 12: The best of Q_v paths, according to the criteria adopted in step 11, is selected and indicated by Q.

STEP 13: Update $T_i = (U_i, S_i)$ by inserting in U_i and in S_i all the nodes and the links of path Q respectively.

STEP 14: Update M by setting $M = M - (M \cap U_i)$. If $M = \emptyset$ and all of the paths P_i belonging to YES have been analyzed go to step 15. If $M = \emptyset$ but not all of the paths P_i belonging to YES have been analyzed go to step 5. If $M \neq \emptyset$ go to step 7.

STEP 15: Group all the obtained trees in the two sets $YESY$ and $YESN$, according to the following rule: $T_i \in YESY$ if $\delta_i \leq \delta_{max}$, otherwise $T_i \in YESN$.

STEP 16: If $YESY = \emptyset$ go to step 17. Otherwise, the tree belonging to $YESY$ having the minimum cost is selected (in case of multiple equivalent trees, the one having the minimum δ is chosen). Output the selected tree and stop.

STEP 17: Select in $YESN$ the tree having the minimum δ (in case of multiple trees, the one having the minimum cost is chosen). The latter is labeled by T_{YN}.

STEP 18: Repeat the loop between steps 6 and 14 for every path $P_i \in NO$ having a δ lower than the fair delay relative to the T_{YN} tree. The new constraint δ'_{max}, replacing the original δ_{max} utilized in (2), is now equal to the fair delay δ_i corresponding to path P_i. Obviously $\delta'_{max} > \delta_{max}$.

STEP 19: Among all the trees generated from steps 17-18, select the one characterized by the minimum fair delay (in case of multiple trees, the one having the minimum cost is chosen). Output the selected tree and stop.

3.3 Performance evaluation

In general, the identification of optimal routing paths from source to destinations, coherent with the QoS targets, can be a problem (16; 22; 30). In fact, when setting up a path, parameters such as delay, bandwidth, loss probability, and jitter must be considered for satisfying QoS requirements. In addition, network efficiency has to be taken into account as well. Multicast routing protocols use various approaches to face this problem. Since it is difficult to satisfy multiple constraints simultaneously, a possible approach would be to define an order in which the various parameters are considered and to evaluate the second parameter, for example, only for those paths considered best with respect to the first parameter. This is the correct point of view to understand the policies adopted in the algorithms proposed and analyzed in this context. Their goal is to incorporate QoS multicasting within efficient routing techniques. The description of the presented algorithms is involved in some cases and it is not simple to identify all the peculiarities of their operation. However, it is evident that they follow the above mentioned strategy to select the best routing paths. Because of this approach, they seem to be quite similar. In spite of this, slight variations present in their logical sequence can play an important role, emphasizing the hidden features of each algorithm. As an example, let us consider the 12-nodes graph of Figure 4(a). In such graph node S is the

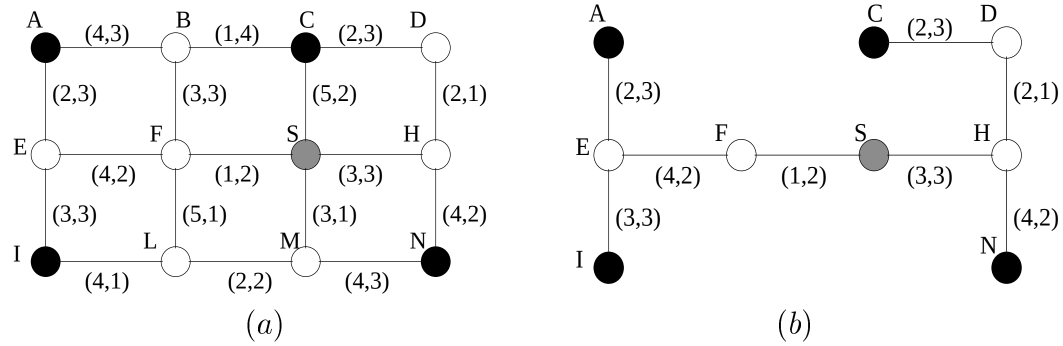


Figure 4. (a) A 12-nodes graph. (b) The CDSAMA-C tree

source, whereas the multicast group is composed by nodes A, C, I and N. Each link is weighted by its corresponding delay and cost, according to the notation (delay, cost). Assuming for the integer parameters the value $k = 3$ and $h = 2$, the CDSAMA-C and CDSAMA-D obtain different multicast trees, drawn in Figures 4(b) and 5(a) respectively. For comparison, the CCDVMA tree is reported too (in Figure 5(b)). The numerical results are then reported

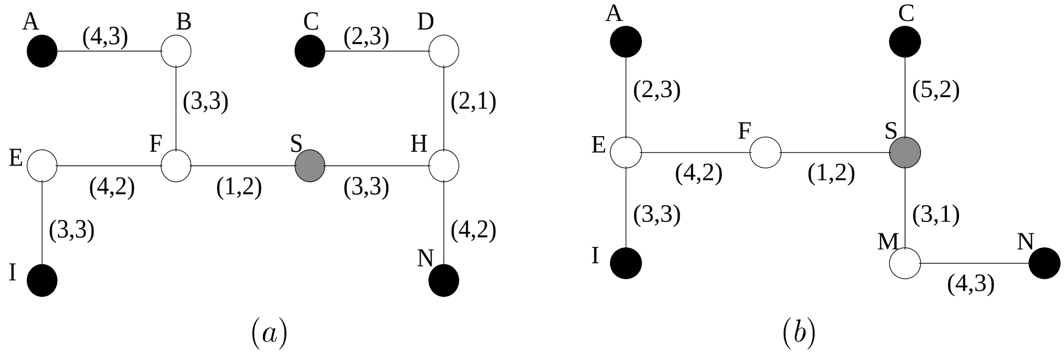


Figure 5. (a) The CDSAMA-D tree. (b) The CCDVMA tree

in Table 1. It is evident that CDSAMA-C and CDSAMA-D have found two trees having the same QoS performance but different cost. On the contrary, CCDVMA is in this case more efficient even though it implies a worse quality in terms of fair delay.

Table 1
Performance of the algorithms applied to the graph of Figure 4(a)

algorithm	Δ	δ	C
CCDVMA	8	3	16
CDSAMA-C	8	1	19
CDSAMA-D	8	1	22

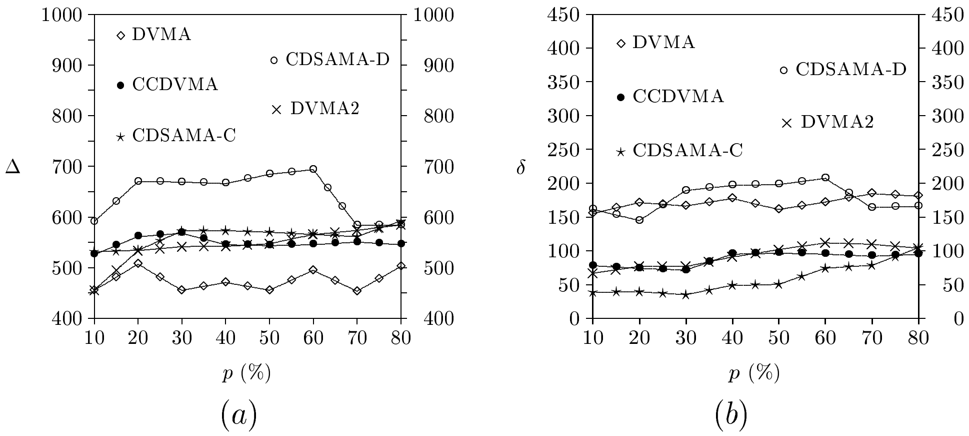


Figure 6. (a) Maximum delay towards 2 destinations in a 20-nodes network. (b) Maximum fair delay towards 2 destinations in a 20-nodes network

Obviously, this is just an example, since a complete comparison has to be performed on a wide variety of different cases, as regards topology, multicast group size, source position, and so on. In this sense, we report here some results obtained via simulation in a more generic context. We tested the algorithms considered in the previous sections, operating in randomly generated topologies. Each node of the network transmits uniformly 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 δ are measured in tens of time slots, whereas C is reported in centesimal units. In Figs. 6, 7 and 8 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:

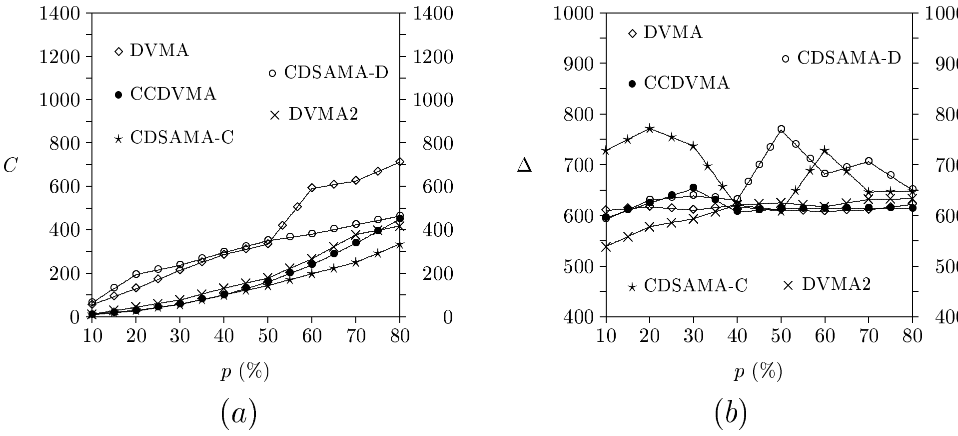


Figure 7. (a) Cost of the tree connecting 2 destinations in a 20-nodes network.(b) Maximum delay towards 4 destinations in a 20-nodes network

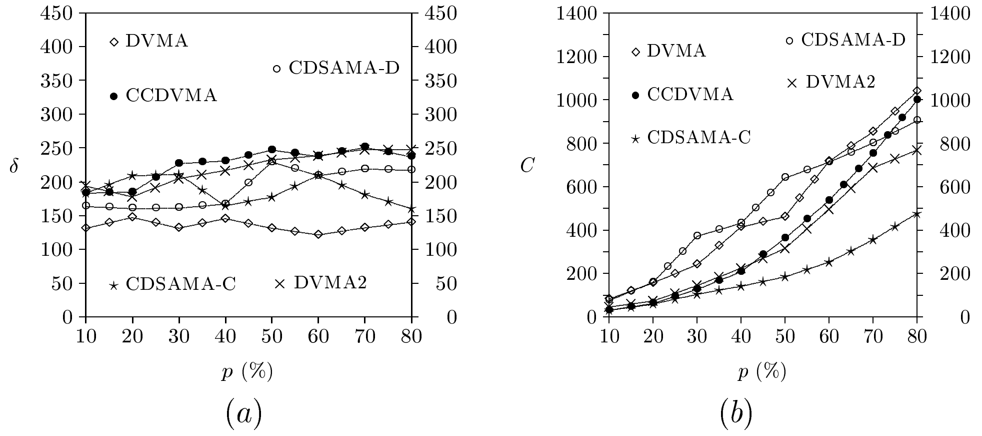


Figure 8. (a) Maximum fair delay towards 4 destinations in a 20-nodes network. (b) Cost of the tree connecting 4 destinations in a 20-nodes network

- 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 but it is preferable for the cost profile and, sometimes, for the differential delay too (see Fig. 6(b));
- CDSAMA-C 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;
- CDSAMA-D is the opposite of CDSAMA-C, 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 simulations 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 (CDSAMA-D excepted). In particular, the corresponding tree cost increases as load gets higher. Among the developed algorithms CDSAMA-D, 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.

4 Distributed Bandwidth Broker

In this section we describe the end-to-end *Call Admission Multicast Protocol* (CAMP) (27), that can be used to ensure bandwidth guarantees to multicast sessions in IP networks, thus providing them with the Premium Service (25). CAMP is scalable, operates on a per-call basis and supports the group membership dynamics. It performs the functionalities of a distributed BB and has been designed to operate within the RTP/RTCP protocol suite (32). CAMP is implemented as a module of the RTP library, which is linked to the applications. It performs the call set-up of an RTP session. In figure 9 we show the relationships between CAMP and RTP. The application calls RTP and passes to it the traffic profile it wants to generate, specified for instance as a `TSpec` (33), and the list of receivers. CAMP supports *homogeneous* recipients, having the same QoS requirements. In section 4.1, we assume that the group membership does not change in the time between the system configuration startup and the end of the transmission. In sections 4.2 and 4.3, we relax this assumption. CAMP exploits the services of RTP and RTCP. RTCP is used to monitor the QoS provided at the recipients. RTP activates CAMP by forwarding it the information received by the application. RTP can be used at the receiver side to appropriately perform the playback of the source transmission. The routers exploit the services of a membership protocol (e.g., IGMP (15)) to discover the group membership of the attached LANs. A multicast routing protocol (e.g., DVMRP (28)) maintains a tree-based multicast infrastructure among the multicast recipients. The multicast routing tree is computed according to the hierarchical routing scheme. CAMP is independent of the underlying multicast protocol. It performs a test to verify whether enough bandwidth is available along the multicast tree to support the specified traffic profile. The test results are reported to the application. To perform the test, all the routers

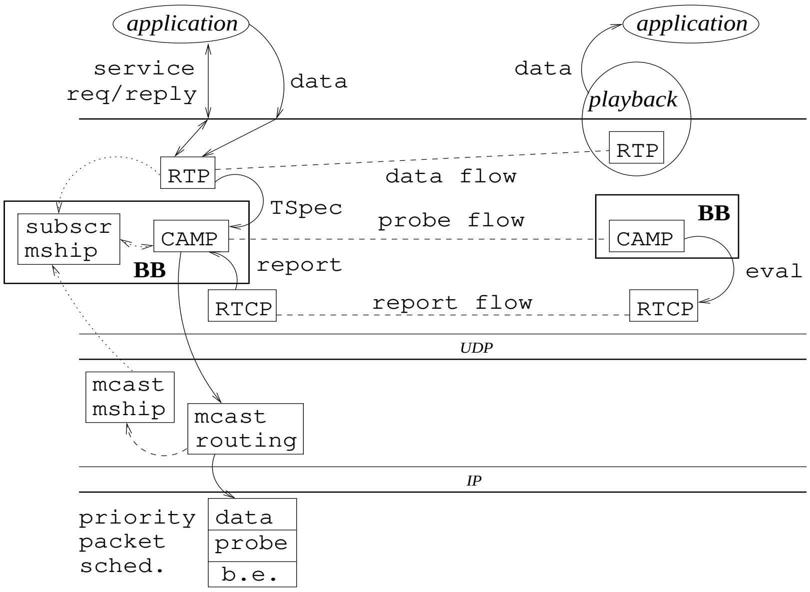


Figure 9. Architecture of a Camp-based end station

must be configured to use a *priority* packet scheduling policy. This requirement is compatible with the implementation of the most part of the current routers.

4.1 Camp *algorithm for static membership*

A multicast session is divided into two phases: the *session set-up* phase, performed by Camp, and the *data transfer* phase, performed through RTP. In the set-up phase, Camp uses the information supplied by the application to test whether the network can guarantee the requested bandwidth. The test is carried out by sending `probe` packets to the destinations, so that the generated probing traffic has the same profile of the specified data traffic. The `probe`s are forwarded along the multicast routing tree. The basic idea consists of differentiating `probe`, QoS data and best effort packets by means of three priority levels. The `probe` packets are marked with a higher priority than the best effort packets and a lower priority than the QoS data packets. This priority assignment ensures that the probing traffic does not affect the existing QoS flows. On the other hand, `probe` packets can drain the available bandwidth for new QoS flows at the expenses of the best effort traffic.

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

If the service is accepted by at least one recipient, then the source switches from the transmission of `probe` packets to the transmission of the data packets generated by the source application, *without discontinuity*. Data packets are forwarded along the pruned tree. In the case of VBR traffic, the CAMP source entity should also generate *dummy* packets in the periods of low source rate, to avoid that the acquired bandwidth is stolen by other traffic.

With the described mechanism, both intra- and inter-domain configurations are automatically performed: all the packet schedulers use the appropriate policy by assumption. The paths along which the multicast traffic must be forwarded are configured by the hierarchical routing service. The resource reservation is ensured by the probing phase thanks to the priority policy: the bandwidth that has been preempted by a given flow during its probing phase cannot be later assigned to other `probe` or best effort packets. Moreover, priority naturally provides flow aggregation. As probing may be started by each host, every host in a domain acts as the BB for the locally generated traffic.

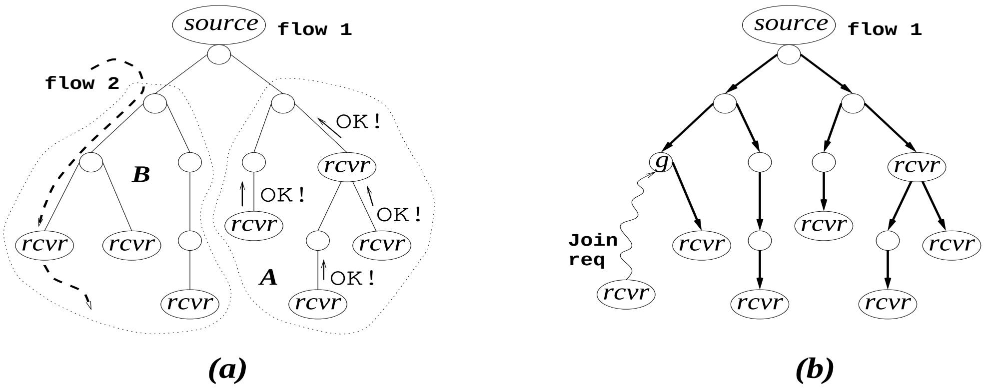


Figure 10. (*a*) Example of concurrent flows. (*b*) Example of dynamic group membership

The knowledge of the group membership is needed because the source *must* wait for the reports from *all* the receivers, before switching to the data transmission. The example of figure 10(a) explains this point. The source generates a QoS flow $flow_1$. If it receives positive reports from a subset of receivers, let us say those in the subtree A, it knows that some destination exists for the data flow. If the source switches to the data transmission before receiving the reports from the receivers in the subtree B, other established QoS flows could be negatively affected. In fact, let us suppose that a flow $flow_2$ exists that traverses the B's branches, and does not leave enough bandwidth for $flow_1$. As a consequence, the receivers in B would eventually send negative reports and prune to refuse $flow_1$. If the switch is performed before receiving all the reports (i.e., before the needed prune operations), the data packets of the aggregate flowing on the subtree B will be dropped, independently of the flow they belong to.

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

4.2 Dynamic membership and related issues

In several applications, such as news services and video broadcasting, the source does not need, or desire, to be aware of the receivers group membership, while receivers are free to join and leave the group dynamically. In this new model, the source announces the multicast session via `sdr` and starts transmitting at the scheduled time if at least one receiver is listening, that is, if the recipient group exists. In all these cases, the CAMP model described above does not apply, because the tree infrastructure that supports the session flow is unable to maintain the bandwidth guarantees in case of topology changes. To understand where the problem arises, let us consider the example shown in figure 10(b), where a receiver wants to join the group when the transmission of a flow $flow_1$ is established. According to the multicast routing protocol, the joining node would graft to an intree router, say g. Anyway, the data packets cannot be forwarded along the new branch, without testing if enough resources are available to accommodate them, for the same arguments discussed before (see figure 10(a)), to avoid incurring in the *neglected reservation subtree problem* (6). The NRSP arises when data packets are forwarded along paths on which no reservation for them has been performed. Since interior routers are QoS-unaware, they can neither detect nor cope with this event. As a consequence, a traffic burst is observed for the aggregate corresponding to the service requested by the packets. This may negatively affect all the packets either belonging to the aggregate or requiring a worse service level.

Hence, the data forwarding must be preceded by the resource reservation. Yet, the probing phase cannot be started at the root, to avoid the duplication of the resource reservation already installed for $flow_1$ on the path from the root to g. On the other hand, if g is an interior router, the probing can neither be started by g, which is QoS-unaware. In fact, $flow_1$ shall be transported through the new branch as part of the proper PHB to which it belongs and, in line with the DS model, the interior routers are unable to identify the subset of packets belonging to $flow_1$, differently from the boundary routers. From those arguments it follows that the proposed mechanism cannot support intra-domain membership changes that affect intree domains when the data transmission is ongoing.

4.3 The CAMP *proxy*

To partially overcome the static membership limitation, in spite of the considerations discussed in section 4.2, we decided to install a CAMP *proxy* in the in-tree boundary routers involved in membership changes. The proxies allow to support inter-domain changes, that is, joining nodes belonging to domains not yet attached to the tree.

When a new node r_{new} wants to join a QoS multicast session S, it is allowed to do so only if the r_{new} domain is not yet intree. If this is the case, r_{new} explicitly joins the proper group of receivers (through, for instance, IGMP report exchange). The r_{new} domain grafts the multicast tree at a boundary router. Let us indicate with f the transmission flow and with G_f the group of the receivers of f. As soon as an intree boundary router g, belonging to the tree T_f for G_f, creates a new downstream interface for T_f, it instantiates the CAMP proxy, say p_f^g, whose lifetime lasts until the set-up phase for the new host terminates. Once activated, the proxy receives the identifier of the new interface and the list of destinations in the downstream domain. It generates, by gathering the information from the local routing table, the data structure of figure 11, where it temporarily maintains the status of the active probing phases. If a join request from a new domain is received while the proxy is active, a new entry of the data structure is created for it. For the arguments discussed in section 4.2, joining nodes in the probing domains are not supported.

probing interfaces	*state*	*destinations*
output interface 1	`data`	{downstream rcvs}_1
...	...	...
...	...	...
...	...	...
output interface n	`probing`	{downstream rcvs}_n

Figure 11. Data structure maintained by a proxy

During its activity the proxy re-marks as `probe` packets all the incoming QoS data packets that should be routed to the probing interfaces. Moreover, the proxy snoops all the packets received from the downstream interfaces. When a RTCP report is received, p_f^g temporarily records it and drops it to prevent its forwarding. When a report has been received from each destination in a probing domain, if at least one positive report exists, then the proxy changes the corresponding entry from the state *probing* to *data*. Otherwise, it removes the entry. The proxy turns off when all the entries in the data structure have been deleted (service refused) or switched to data transfer. This simple mechanism solves the problem due to a dynamically changing inter-domain tree topology and achieves the following goals:

(1) the "`probe`" packets allow to evaluate the resource availability only along the new branch;
(2) the new destinations immediately start receiving the real data flow, although possibly with a lower quality than required;
(3) the data sent to the new destinations do not affect previously established flows traversing the new branch, as the data packets are marked and treated as `probe` packets;
(4) the membership is hidden to the source. Anyway, it must be known by the boundary router because of the same arguments discussed in section 4.1.

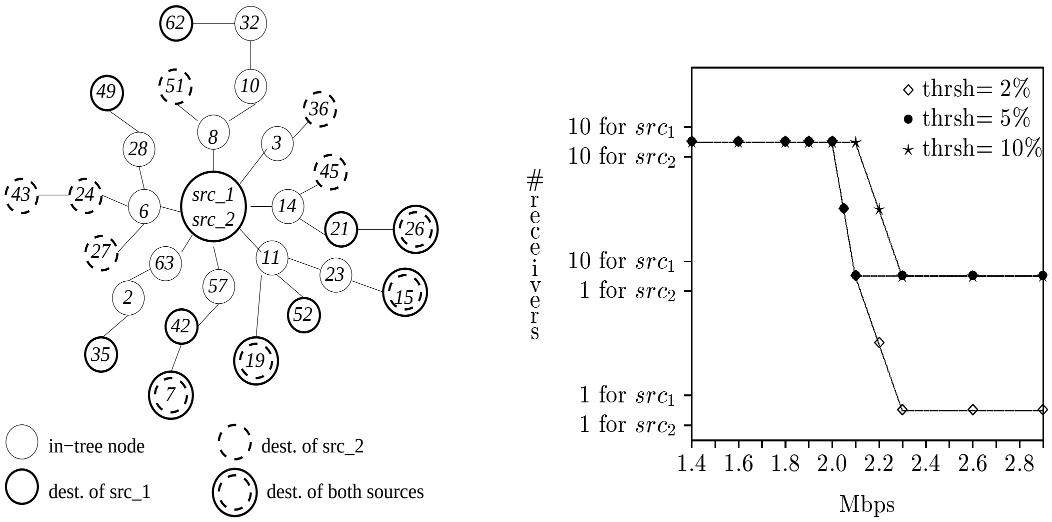
To show that the proxy mechanism is in practice very general, let us consider the system condition when the multicast session is initiating. At that time, we assume that a certain amount of recipients joined the multicast tree (in the trivial case of no recipients the transmission would not be initiated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

The implementation of the proxies requires drastic changes with respect to the current router architecture. For the moment, those changes concern the boundary routers only, as we assume that membership changes are uniquely supported at the inter-domain level. According to the diff-serv model, it is probable that the boundary routers will be dedicated machines with peculiar capabilities to perform traffic classification, shaping and remarking. So far, in our knowledge, no network device exists that automatically performs those functions. In the design and implementation of future devices, proxy capabilities may as well be provided. In the meantime, only static groups can be supported, according to the planned initial phase of the diff-serv model deployment (34; 5).

We have implemented the architecture shown in figure 9, in the frame of the NS-2 simulation package (17), to evaluate how CAMP behaves when several QoS flows compete with the best effort traffic for the available bandwidth. We considered the DVMRP multicast routing protocol and we embedded a real RTP implementation (31) into the RTP template of NS-2. So far, we considered static group membership. The multicast tree is formed before the transmission starts. The sources know the group of recipients. During the probing phase, the recipients compare the received rate with the source rate: if the difference is below a tolerated threshold, a recipient sends a positive report. We performed measures for different thresholds. The decision is sent within the first RTCP report a destination generates after the reception of a number of `probe` packets, i.e. of samples, sufficient to ensure an accurate measure of the available bandwidth by covering the rate of the slowest traffic source. We performed simulations with different sample sizes.

The first set of simulations has been performed with a meshed network of 64 nodes, connected by optical links of 2 Mbps bandwidth and variable length in the range 50 to 100 Km. Background, best effort traffic is uniformly distributed all over the network; it is generated by 32 unicast connections among randomly chosen pairs of nodes. Best effort sources generate CBR traffic with a 0.66 Mbps rate, so that about 33% of the network resources is used. The size of best effort, `probe` and data packets is 512 bytes. Two CBR sources are located on the same node. The first source generates traffic at the rate of 1 Mbps; the rate of the second source assumes different values in the range 0.4 Mbps to 1.9 Mbps. The probing phase of the second source starts 1.5 sec. after the start of the first source, so that the two set-up phases partially overlap. The recipients decide after the reception of 500 `probe` packets; hence, the length of the set-up phase depends on the source rate. By performing simulations with different group cardinalities for the two groups we observed that the performance is almost independent of the group cardinality. In the following, we only report the results we obtained when both groups have 10 destinations. Groups partially overlap: the simulated topology restricted to the multicast routing trees for the two groups is shown in figure 12(a).

Simulations indicate that CAMP effectively performs the call admission control. In fact, until the sum of the source rates is lower than the link capacity, the recipients in both groups accept the transmissions and receive at the correct data rate. When the rate of the second source is higher than 1 Mbps, the source transmission can be accepted only by receivers linked through separate branches to the multicast trees (nodes 35 and 36 in figure 12(a)). When the multicast trees of the two groups share the same link, only the destinations in one group shall be able to accept the transmission. CAMP correctly identifies



(a) (b)

Figure 12. (a) Overlapped multicast trees for the QoS sources. (b) Overall number of accepting recipients for both groups, for different threshold values and $|G| = 10$

the receivers able to receive at the proper data rate, removes the others from the group, and performs this task with the required fairness. In fact, it ensures that the transmission that started the probing phase earlier would not be negatively affected by the second group, and, when in competition, would receive the bandwidth guarantees as shown in figure 13. In figure 12(b) we report the number of recipients of both groups that accepted the transmission, with respect to the traffic load. In figure 13(a) we show with a solid line the rate received at the destinations that accept the service. The dashed line represents the average rate obtained during the set-up phase by considering all the receivers, and also represents the data rate that would be observed at the destinations without the call admission control. As shown in figure 12(b), by setting the threshold to 2%, 5%, 10% of the source rate, CAMP is able to properly provide the bandwidth guarantees to all the receivers in both groups until the data rate does not saturate the link capacity. At a higher request of bandwidth, and threshold at 5% and 10%, all the receivers in the first group accept the transmission while, in the second group, only the destinations on separate branches accept the transmission (in the simulation, only 1 receiver: node 36). As confirmed by the results of figure 13(a), the receivers able to accept the transmission receive the adequate bandwidth guarantees to receive at the correct data rate. When the threshold is set too low, 2%, the number of receivers able to accept the transmission becomes very low due to the observed delay and jitter, as discussed below. The acceptance threshold should be chosen according to the semantics of the source application, that is, it should be no greater than the maximum packet loss probability that can be tolerated so that the information does not become useless.

In figure 13(b) we report the observed end-to-end delay, while in figure 14(a) we

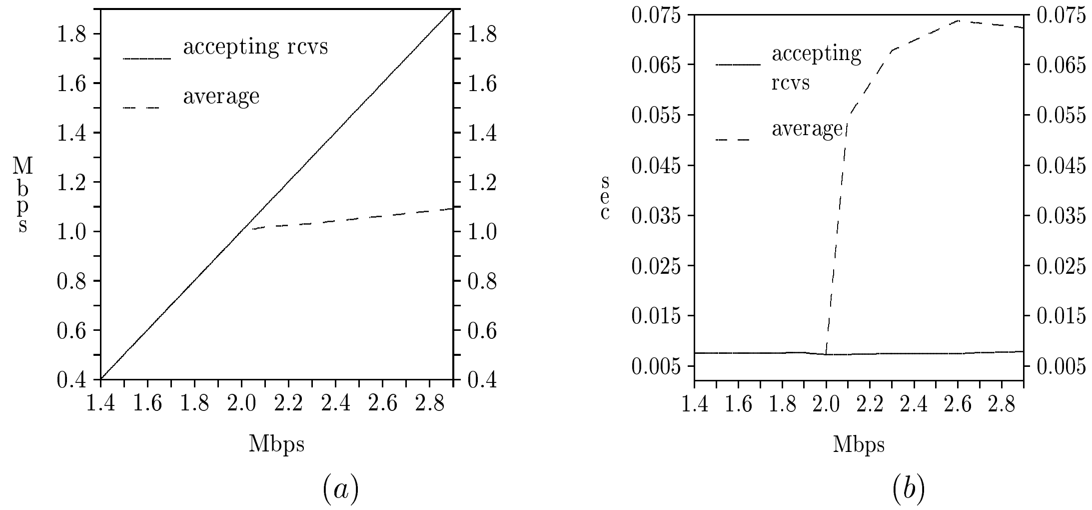


Figure 13. (a) Average received rate for the second group of recipients vs. offered load, for $|G| = 10$ and threshold value 5%. (b) Average end-to-end delay for the second group of recipients vs. offered load, for $|G| = 10$ and threshold value 5%

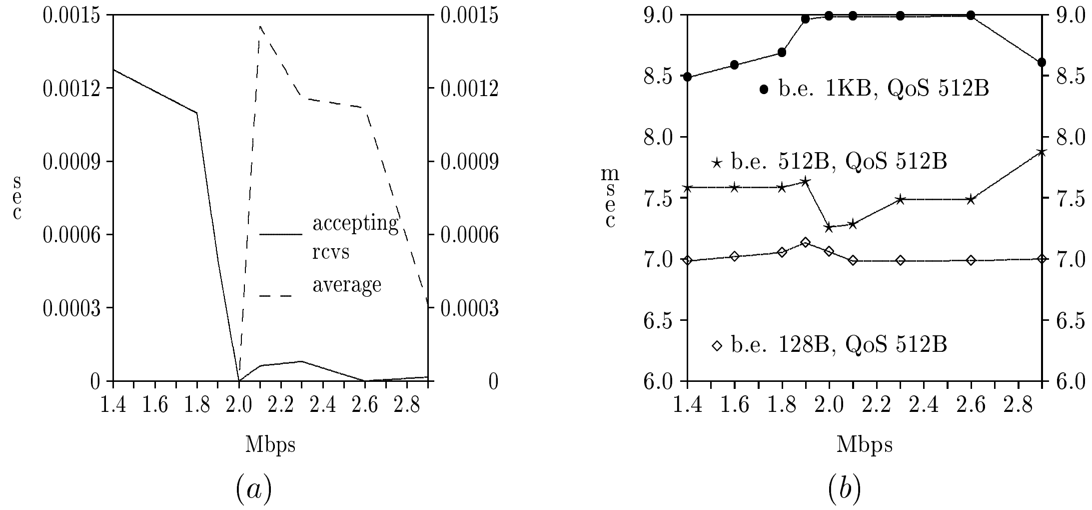


Figure 14. (a) Average jitter for the second group of recipients vs. offered load, for $|G| = 10$ and threshold value 5%. (b) Average end-to-end delay for the second group of recipients vs. packet size, for $|G| = 10$ and threshold value 5%

report the jitter. The jitter is the variance on the packet inter-arrival time and has been computed according to the algorithm given in the RTP specification (32).

The end-to-end delay is almost constant before saturating the links, and it remains unchanged for the accepting destinations with congested links. This indicates that the flows characteristics are preserved from source to destination, independently of the network load and the group cardinality. The measure of the jitter shows the effects of the presence of best effort packets at the core routers. As expected, the priority mechanism alone is not sufficient to ensure delay and jitter control at the destinations. As a consequence, the jitter decreases together with the decreasing of the amount of best effort packets competing to use the link resources. Jitter approaches 0 when both the QoS sources generate at rate 1 Mbps (in this case, there is no space left for best effort traffic). At higher QoS rate some destination has to refuse the service (see figure 12(b)), thus allowing that a few best effort packets still succeed in accessing the network. As shown by figure 14(a), their effect is marginal over the links where the accepted traffic is forwarded.

Although the proposed admission control mechanism has proved to be effective to provide bandwidth guarantees, the system characteristics may affect other QoS parameters. To highlight the impact of the best effort traffic over the QoS, we performed simulations with different packet sizes (the `probe` packets always have the same size as the QoS data packets). In figure 14(b) we report the end-to-end delay observed by the QoS packets of the second source, that compete with best effort traffic generated as before. The delay observed by the data packet size depends on the best effort packet size. As the links cannot be preempted once a packet transmission is ongoing, QoS packets arriving at a node could have to wait at most for a best effort packet transmission time before gaining the link, although they have the highest priority. The impact of the measured delays over the received rate is however always negligible.

The jitter measured at the recipients may be affected both by the distance of a recipient from the source, and by the mix of flows traversing the path toward the recipient. In figure 15(a), we show the average jitter measured in a network having a tree topology, such that the two QoS sources are located in the root node, to which three recipients are grafted at distances of respectively 1, 5 and 10 hops. The tree links have 2 Mbps bandwidth each. Only the QoS traffic is generated. While for low QoS rate of the second source the jitter is independent of the distance from the root, when the traffic load approximates the network saturation point, queueing, transmission and propagation delays alter the regular traffic profile.

We as well investigated how transmissions of different rates interact. We compared the performance achieved by (**I**) 2 sources with rate 0.9 Mbps; (**II**) 5 sources with rate 0.36 Mbps; (**III**) 10 sources with rate 0.18 Mbps; and (**IV**) 1 source with rate 0.9 Mbps and 5 sources with rate 0.18 Mbps. A group of 10 recipients receives from all the destinations; the threshold value is 5%. While the measured rate is always correct, the jitter depends on the flow mix (figure 15(b)). Although the traffic load is the same under all the configurations, many sources with low rate have a lower probability of competing to access the links in the intermediate routers than a few sources with high rate. At the same time, a high number of sources with low rate produces an overall traffic that is less regular than that produced by a single source. Those irregularities can affect other sources that compete to access the same links (configuration IV). At the recipients, the jitter can be annihilated by exploiting the RTP

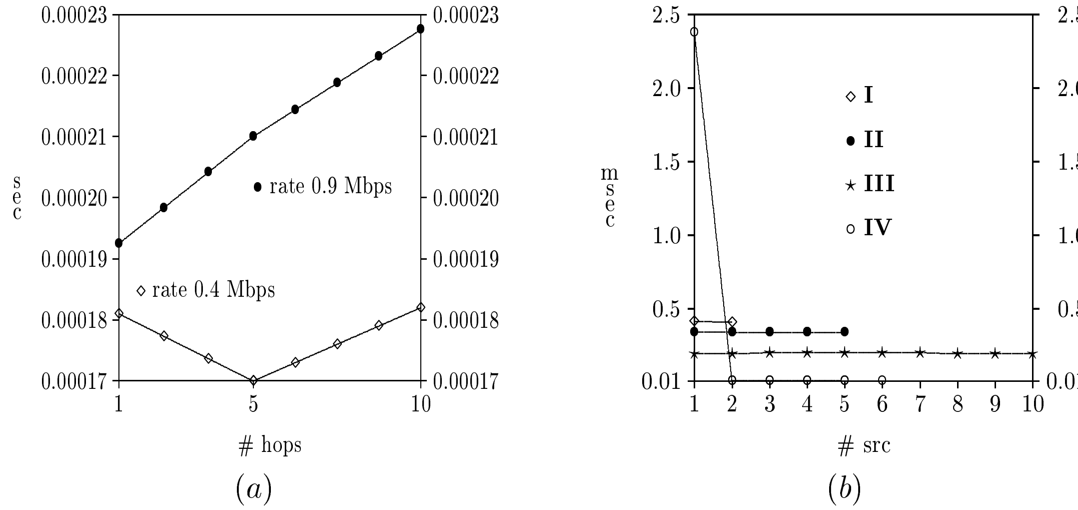


Figure 15. (*a*) Average jitter vs. distance from the source with threshold value 5%. (*b*) Average jitter for different mix of flows, with threshold value 5%

service to deliver the received packets at a constant rate. On the other hand, in some environments (e.g., with congested network, low threshold, or in systems larger than that we studied in our simulations), the observed delay and jitter behaviours could correspond to a lower service acceptance probability. These considerations deserve further investigation.

5 Concluding remarks

In this paper we propose some policies to implement bandwidth broker agents, to support multicast QoS in the diff-serv model. We discuss the performance results obtained by implementing those policies in a simulation environment.

The algorithms proposed in section 3 have been studied and developed with the goal of supporting real-time multi-user applications among remote users. That is why we focused our attention, in particular, on the time alignment of end terminals which, in the context considered, need relatively high mutual synchronization. On the other hand, the efficiency of the multicast connection cannot be neglected from the perspective of real use of the solutions presented here. The results accumulated so far seem to demonstrate good agreement with our expectations. At present, two refinements of our work have been initiated:

- inclusions of suitable tree re-fitting techniques for enabling join/leave operations on the multicast group, during a multi-user communication session;
- inclusions of differentiated QoS requests, for different subsets of users belonging to the same multicast group.

As regards the latter aspects, we will remove the hypothesis of homogeneous recipients, by investigating the possibility of distributing different bandwidth resources (strictly related to link cost) among users, through modulation of the corresponding bounds, within the same tree. Similarly, the maximum delay could be negotiated as well.

The CAMP library proposed in section 4 adopts an innovative approach, to support dynamic changes in the group of recipients. CAMP proxies are dynamically created in the routers that must take care of those changes. They perform the updates in the system configuration required by the membership changes, and then they are destroyed. We implemented the described mechanism in the frame of the NS-2 simulation package. The achieved results are promising: the devised approach effectively performs admission control. The accepted services have the requested QoS guaranteed. Yet, further investigation has to be carried out concerning the interactions amongst several concurrent transmissions and their impact on the probability of successful service establishment.

So far, proxies can support dynamic multicast membership under the assumption that all the recipients grafted to the tree receive the same transmissions. We are extending their capabilities so that heterogeneous destinations, willing to accept different sets of data flows, can be supported. Similarly, the mixing and translation functionalities of RTP could be embedded into the proxies. In the near future, we will be able to perform experiments with the described architecture in the frame of a testbed that we are currently deploying over the departmental network (23).

References

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

[2] T. Bates, Y. Rekhter, R. Chandra, D. Katz, Multiprotocol Extensions for BGP-4, *RFC 2858* (Jun. 2000). Work in progress.

[3] G. Bianchi, F. Borgonovo, A. Capone, L. Fratta, C. Petrioli, PCP: an End-to-End Measurement-Based Call Admission Control for Real-Time Services over IP Networks, in: *Proc. International Workshop on QoS in Multiservice IP Networks, Lecture Notes in Computer Science, vol. 1989*, Springer, Berlin, Jan. 2001, 391.

[4] G. Bianchi, N. Blefari-Melazzi, A Migration Path for the Internet: From Best-Effort to a QoS Capable Infrastructure by Means of Localized Admission Control, in: *Proc. International Workshop on QoS in Multiservice IP Networks, Lecture Notes in Computer Science, vol. 1989*, Springer, Berlin, Jan. 2001, 422.

[5] S. Blake, D. Black, M. Carlson, E. Davies, Z. Wang, W. Weiss, An Ar-

chitecture for Differentiated Services, *RFC 2475* (Dec. 1998). Work in progress.

[6] R. Bless, K. Wehrle, IP Multicast in Differentiated Services Networks, *Internet Draft, draft-bless-diffserv-multicast-01.txt* (Nov. 2000). Work in progress.

[7] A. Borella, G. Cancellieri, Improving the Cost/Performance Ratio in Multicast Communications, in: *Proc. NOC 2000* (Jun. 2000) 187–190.

[8] A. Borella, G. Cancellieri, Remote Time-Alignment of Interactive Services through Efficient Multicast Algorithms, in: *Proc. ECUMN 2000* (Oct. 2000) 239–246.

[9] A. Borella, G. Cancellieri, E. Pagani, G.P. Rossi, Quality-of-Service Guarantees for Multicast Traffic in Heterogeneous Multi-Service Networks, in: *Proc. International Workshop on QoS in Multiservice IP Networks, Lecture Notes in Computer Science, vol. 1989*, Springer, Berlin, Jan. 2001, 97–112.

[10] F. Borgonovo, A. Capone, L. Fratta, M. Marchese, C. Petrioli, PCP: A Bandwidth Guaranteed Transport Service for IP networks, in: *Proc. IEEE ICC'99* (Jun. 1999).

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

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

[13] S. Bradner, Internet Protocol Quality of Service Problem Statement, *Internet Draft, draft-bradner-qos-problem-00.txt* (Sep. 1997). Work in progress.

[14] B. Cain, Connecting Multicast Domains, *Internet Draft, draft-ietf-mboned-mcast-connect-00* (Feb. 2000). Work in progress.

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

[16] S. Chen, K. Nahrstedt, An overview of quality of service routing for next generation high-speed networks: problems and solutions, in: *IEEE Network* (Nov./Dec. 1998) 64–79.

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

[18] D. Farinacci, Y. Rekhter, D. Meyer, P. Lothberg, H. Kilmer, J. Hall, Multicast Source Discovery Protocol (MSDP), *Internet Draft, draft-ietf-msdp-spec-06.txt* (Jul. 2000). Work in progress.

[19] B. Haberman, G.N. Rouskas, Cost, Delay and Delay Variation Conscious Multicast Routing, in: *Proc. INFOCOM 2000* (2000).

[20] M. Handley, *The sdr Session Directory: An Mbone Conference Scheduling and Booking System* (Apr. 1996), `http://mice.ed.ac.uk/mice/archive/sdr.html`.

[21] Internet2 QoS Working Group Draft, QBone Architecture (v1.0) (Aug.

1999), http://www.internet2.edu/qos/wg/papers/qbArch/1.0/draft-i2-qbone-arch-1.0.html. Work in progress.

[22] A. Iwata, R. Izmailov, D. Lee, B. Senegupta, G. Ramamurthy, H. Suzuki, ATM routing algorithms with multiple QOS requirements for multimedia internetworking, *IEICE Transactions on Communication*, **vol. E79-B, n. 8** (Aug. 1996) 999–1007.

[23] D. Maggiorini, E. Pagani, G.P. Rossi, A Testbed Environment for the Performance Evaluation of Modular Network Architectures, in: *Proc. 1st IEEE European Conference on Universal Multiservice Networks (ECUMN 2000)* (Oct. 2000) 283–292.

[24] K. Nichols, S. Blake, F. Baker, D. Black, 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.

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

[26] E. Pagani, G.P. Rossi, A Functional Architecture for End-to-end Quality-of-Service in a Multi-domain Network, in: *Proc. 1st IEEE European Conference on Universal Multiservice Networks (ECUMN 2000)* (Oct 2000) 20–34.

[27] E. Pagani, G.P. Rossi, D. Maggiorini, A Multicast Transport Service with Bandwidth Guarantees for Diff-Serv Networks, in: *Proc. International Workshop on QoS in Multiservice IP Networks, Lecture Notes in Computer Science, vol. 1989*, Springer, Berlin, Jan. 2001, 129–140.

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

[29] G.N. Rouskas, I. Baldine, Multicast Routing with End-to-end Delay and Delay Variation Constraints, *IEEE Journal on Selected Areas in Communications* **vol. 12, n. 3** (Apr. 1997) 346–356.

[30] H.F. Salama, D.S. Reeves, Y. Viniotis, Evaluation of multicast routing algorithms for real-time communication on high-speed networks, *IEEE Journal On Selected Areas In Communications* **vol. 15, n. 3** (Apr. 1997) 332–345.

[31] H. Schulzrinne, *RTP: About RTP and the Audio-Video Transport Working Group*, http://www.cs.columbia.edu/~hgs/rtp/ .

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

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

[34] B. Teitelbaum, P. Chimento, QBone Bandwidth Broker Architecture, *Internet 2 QoS Working Group Draft* (Jun. 2000), http://qbone.internet2.edu/bb/. Work in progress.

[35] L. Zhang, R. Braden, S. Berson, S. Herzog, S. Jamin, Resource reSer-

Vation Protocol (RSVP) - Version 1 Functional Specification, *RFC 2205* (Sep. 1997). Work in progress.

Andrea Borella was born in Bologna, Italy, in 1962. He received his laurea degree in Electronic Engineering in 1987 from the University of Bologna (summa cum laude). Until 1988 he worked at the Radioastronomy Institute of CNR (National Research Council). In 1989 he joined the Department of Electronics and Automatics of the University of Ancona, where he received his PhD in Electronic Engineering in 1993. His research activity essentially concerns performance evaluation of high speed networks: topologies, architectures, protocols and routing techniques; traffic modeling and network simulation. He is member of IEEE.

Giovanni Cancellieri was born in Florence in 1952. He received his Laurea degree in Eletronic Engineering and in Physics in 1976 and 1978 respectively, both from the University of Bologna. He is currently a Full Professor at the Department of Electronics and Automatics of the University of Ancona, where he leads the Communications Research Group. His research interests are in the field of fiber optics, optical signal transmission and processing, and coding. He is member of IEEE, EATA and AEI. E-mail: cangio@popcsi.unian.it

Elena Pagani received her Ph.D. degree in Computer Science from the University of Milan in 1999. She is currently an Assistant Professor at the Department of Computer Science of the University of Milan. Her research interests include multicast QoS in Internet, wireless networks, distributed systems and performance evaluation. E-mail: pagani@dsi.unimi.it

Gian Paolo Rossi is currently a Full Professor at the Department of Computer Science of the University of Milan, where he is leading the Computer Networks Laboratory. His research interests include multicast QoS, mobile computing and networks, and distributed systems. He is member ACM and IEEE. E-mail: rossi@dsi.unimi.it