Call Admission and Resource Reservation for Multicast Sessions *

Victor Firoiu
vfiroiu@cs.umass.edu

Don Towsley
towsley@cs.umass.edu

Department of Computer Science
University of Massachusetts
Amherst, MA 01003-4610 USA

Abstract

Many multicast applications, including audio and video, require quality of service (QoS) guarantees from the network. Hence, multicast admission control and resource reservation procedures will be needed. In this paper we present a general framework for admission control and resource reservation for multicast sessions. Within this framework, efficient and practical algorithms that aim to efficiently utilize network resources are developed. The problem of admission control is decomposed into several subproblems that include: the division of end-to-end QoS requirements into local QoS requirements, the mapping of local QoS requirements into resource requirements, and the reclaming of the resources allocated in excess. These are solved independently of each other yielding a set of mechanisms and policies that can be used to provide admission control and resource reservation for multicast connection establishment. The resource allocation algorithms we consider specifically accommodate receiver heterogeneity (in both end-to-end and per-hop QoS requirements) by reserving necessary and sufficient resources for a multicast session. An application of these algorithms in the context of packetized voice multicast connections over the Mbone is provided to illustrate their applicability.

1 Introduction

The increasing accessibility of IP-multicast on the Internet has spurred a rapid growth in the number of multicast applications using the Internet. These include audio (vat [14], NeVoT [17]) and video (nv [12]) which, being constrained to provide smooth play-out at the receiver, require connection-oriented services and the allocation of sufficient network resources in order to guarantee the desired play-out quality. Unfortunately, such services are currently not provided by the Internet and the play-out quality for such applications has been quite variable.

In this paper we address the problem of connection setup for such multicast applications that require quality of service (QoS) guarantees on end-to-end delay or loss probability. A solution to this problem requires establishing a multicast tree (routing), checking whether or not the application can be admitted on that tree (call admission), and reserving resources. We present a general framework for developing algorithms for performing call admission and reserving the resources required to provide a desired quality of service (QoS) to a multicast connection. Within this framework, we also develop algorithms to solve these problems. The approach to resource reservation is based on the division of the end-to-end QoS requirement for each receiver into local QoS requirements at each of the links on the path from the source to that receiver. A key result of our work are algorithms which reclaim resources reserved in excess due to the fact that two or more receivers may impose different local QoS requirements on a path segment that they share.

Much of the previous work on call admission has focussed on the development of mathematical models for providing performance bounds, with applications to call admission of unicast connections (see[5, 9, 16]). Resource allocation has been another topic of research in recent years and various protocols have been developed: RSVP ([21]), ST-II ([18]), (also see [10, 1, 3]). Although RSVP and ST-II support multicast applications, they merely provide mechanisms, but not policies, for performing resource allocation.

Route establishment is an important part of connection establishment. However, we do not consider it in this paper for several reasons. First, it has been studied extensively in recent years (see [6, 2, 7, 8, 15, 19]). Second, we believe that a good solution to the combined problem of routing and call admission is one that focuses on each separately. Several papers have considered the combined problem of

*This material is based upon work supported in part by the National Science Foundation under Grant NCR-95-08274. Any opinions, findings, and conclusions or recommendations expressed in this material are those of the authors and do not necessarily reflect the views of the National Science Foundation.

route establishment and call admission of multicast sessions, [15, 19]. However, these works have focussed on solving a *static optimization* problem where all of the multicast sessions are known. A solution to this problem then provides the routes and resource allocation that minimizes some cost function that includes link cost and delay. The solution is usually obtained through integer programming. This problem reduces to the well known Steiner tree problem that has proven to be NP-complete. Our conclusion that the two problems should be decoupled is also supported by observations in [19] that the cost of the multicast tree constructed from shortest path multicast connections is close to the minimum cost solution. Similar observations were also made in [7].

Our approach to the problem of multicast connection establishment is as follows: first route the multicast connection (determine the multicast tree) using one of the existing algorithms and then do resource allocation (with admission control) and resource reclaiming on the multicast tree. We address these latter two problems. The algorithms are developed to solve a static multicast problem (where all receivers are known before the session's setup). An extension to solve a dynamic multicast problem (where receivers can join or leave anytime during the life of the session) is presented in [11]. Distributed algorithms for the receiver oriented multicast connection establishment is the subject of a future work.

In Section 2 we give a formal statement of the problem. A decomposition of the problem is found in Section 3 along with a number of alternative solutions to each subproblem. Section 4 illustrates the effectiveness of our algorithms in the context of packetized voice multicast applications with a packet loss constraint running on a network in which generalized processor sharing policies are used as the link schedulers. The paper concludes with a discussion of some of our assumptions along with a description of work to be done in the future.

2 Problem description

In this section we formalize the problem of call admission and resource allocation for multicast sessions. We begin with some notation:

- A network is represented by a directed graph $G(V, E)$ with weights associated with links (edges) $(R_l)_{l \in E}$, $R_l \geq 0$ corresponding to available resources; a link l from A to B in G is denoted by $A \xrightarrow{l} B$.
- $\mathcal{M} = (S, \mathcal{D}, (Q(S, D))_{D \in \mathcal{D}})$ denotes a multicast session where S is the source node, $\mathcal{D} = \{D^1, ..., D^N\}$ is a set of N destination nodes and $Q(S, D)$ is the end-to-end QoS requirement for the (unicast) connection from S to $D \in \mathcal{D}$.
- $\mathcal{T} \subseteq G(V, E)$ is a directed tree in G (the multicast tree) corresponding to routes from S to all $D \in \mathcal{D}$.
- If a path exists from A to B in $\mathcal{T}$, denoted by (A, B) then $\mathcal{L}(A, B)$ is the set of links, and $\mathcal{N}(A, B)$ is the set of nodes (including A and B) on that path. Note that if the path exists in $\mathcal{T}$, it is unique. In the rest of this paper we will consider only paths in $\mathcal{T}$.

The static multicast call admission problem addressed in this paper is the following:

> *Given a network G, a multicast session $\mathcal{M}$, and a multicast tree $\mathcal{T}$, based on the available network resources $(R_l)_{l \in E}$, determine whether $\mathcal{M}$ can be admitted on route $\mathcal{T}$ and, if so, reserve the necessary network resources.*

The dynamic multicast problem is a variation of the above where receivers can join or leave anytime during the life of the session. In this paper we address the static multicast problem. The dynamic multicast problem is addressed in [11].

In solving the above problem we make the following assumptions:

- The multicast tree $\mathcal{T}$ is known.
- The QoS metric considered is additive: the end-to-end guarantee on a path can be expressed as the sum of the link guarantees on that path.
- The resource needed by a session on a link is a fraction of the total link bandwidth.
- Link resource requirements increase as link QoS requirements become more stringent.
- Each node (router) includes a switch which is responsible for packet forwarding on all outgoing links, and a control processor which is responsible for resource bookkeeping and admission control.

3 Structured Solution

We present a two phase algorithm for performing call admission of multicast sessions in a network. First, an allocation phase determines whether there are sufficient resources along the paths within $\mathcal{T}$ to the destinations in order to guarantee the end-to-end QoS requirements of $\mathcal{M}$. If the QoS requirements can be guaranteed, $\mathcal{M}$ is admitted and the allocation phase performs an initial allocation of resources to satisfy those requirements. A second, reclaim phase is concerned with releasing some of the resources allocated in $\mathcal{T}$ by taking advantage of situations where different destinations share a path segment and require different amounts of resources on that segment.

The allocation phase algorithm includes the following functionally independent components:

1. A mechanism to relate the local QoS requirement to the resource to be allocated at the link (Section 3.1);
2. A policy to map the end-to-end QoS requirement for a single destination into local QoS requirements (Section 3.2).

The allocation phase algorithm is described in Section 3.3 and the reclaim phase algorithm in Section 3.4.

3.1 Mapping local QoS requirements to link resources

The mapping between local QoS requirement and resources at a link depends on the nature of the QoS (loss probability, delay), the workload characteristics of the session (e.g. linear bounded arrival process (LBAP) [16], exponentially bounded burstiness (EBB) [20], on/off Markov fluid [13, 9]) and the link scheduling policy (First Come First Served (FCFS), Generalized Processor Sharing (GPS) [16], Earliest Deadline First (EDF) [10]). We assume that the link scheduling policy exhibits the following two properties. First, the link scheduling policy supports sessions requiring different local QoS guarantees at a link. A second, highly desirable (but not mandatory) property is that a newly accepted session does not affect any of the pre-existing sessions' QoS guarantees at a link. The Generalized Processor Sharing (GPS) link scheduling policy exhibits both properties, and will be part of our example in Section 4.

Henceforth, we assume the existence of the following functions:

$$F_l(Q) \mapsto T \ , \ H_l(T) \mapsto Q$$

where Q is the local QoS requirement at link l and T is the amount of resources needed at l to guarantee Q, and H_l is the inverse function of F_l ($H_l = F_l^{-1}$).

3.2 Mapping end-to-end QoS to local QoS

In order to solve the problem of allocating resources at the links given an end-to-end QoS requirement, we need to first divide the end-to-end QoS requirement into local QoS requirements (such that a session meeting every local QoS requirement will also meet the end-to-end QoS requirement) and then determine the resources required at each link in order to achieve the local QoS using the function F_l. QoS division is implemented in the following procedure:

$$\text{COMPUTE\_QOS\_PATH}(P, V_P, Q; (Q_l)_{l \in \mathcal{L}(P)}) \quad (1)$$

which takes as input a path $P = (A, D)$ in $\mathcal{T}$, with attribute V_P, and the end-to-end QoS requirement Q for P. It outputs the local QoS requirements Q_l for each $l \in \mathcal{L}(P)$ such that $\sum_{l \in \mathcal{L}(P)} Q_l \leq Q$ and $F_l(Q_l) \leq R_l \quad \forall l \in \mathcal{L}(P)$ where R_l is the amount of available resources at l. In many cases we need the value Q_l for only one link $l \in \mathcal{L}(P)$ or a value Q_P that characterizes the QoS division on P (see the end of Section 3.2.3 for the use of Q_P). Given a path P with attribute V_P, a link l with attribute V_l, and the QoS requirement Q for P as input, then the procedure:

$$\text{COMPUTE\_QOS\_LINK}(P, l, V_P, V_l, Q; Q_l, Q_P) \quad (2)$$

has in some cases (see Section 3.2.2) lower computational complexity than COMPUTE_QOS_PATH. The procedures COMPUTE_QOS_PATH and COMPUTE_QOS_LINK and path and link attributes are described in Section 3.2.2 for several QoS division policies.

3.2.1 The calculus of QoS division

Let $A \xrightarrow{l} B \xrightarrow{m} C$ be two adjacent links in network G. If the QoS is packet loss probability, then an upper bound on the end-to-end loss probability is:

$$\Pr(\text{loss}(A, C)) < \Pr(\text{loss}(A, B)) + \Pr(\text{loss}(B, C))$$

In the case that $\Pr(\text{loss}(A, C))$ is small ($< 10^{-2}$), this is a tight bound. In this case we write $Q(A, C) \approx Q_l + Q_m$.

If the QoS is a constraint on packet delay, then

$$\text{max\_delay}(A, C) = \text{max\_delay}(A, B) + \text{max\_delay}(B, C)$$

and we write $Q(A, C) = Q_l + Q_m$.

3.2.2 End-to-end QoS division policies

We describe two policies for dividing end-to-end QoS requirements among the links on the end-to-end path. The first policy attempts to divide the QoS requirements evenly among the links, whereas the second allocates more stringent QoS requirements to links having more available resources.

Even division policy This policy allocates equal shares of the end-to-end QoS among links on a path, wherever possible. We begin by assuming that links have sufficient resources to provide the local QoS computed by the Even division policy. In COMPUTE_QOS_PATH$(P, V_P, Q; (Q_l)_{l \in \mathcal{L}(P)})$, the end-to-end QoS requirement Q on path P is divided into $\mid \mathcal{L}(P) \mid$ equal parts: $Q_l = Q / \mid \mathcal{L}(P) \mid, \ \forall l \in \mathcal{L}(P)$, using the path attribute $V_P = \mid \mathcal{L}(P) \mid$. In COMPUTE_QOS_LINK$(P, l, V_P, V_l, Q; Q_l, Q_P)$ the outputs are $Q_l = Q_P = Q / \mid \mathcal{L}(P) \mid$ for a specified $l \in \mathcal{L}(P)$. In this computation only the path attribute is needed ($V_P = \mid \mathcal{L}(P) \mid$ and $V_l = \emptyset$).

The case where one or more links has an insufficient amount of resources to accommodate an even division of QoS is considered in [11].

Proportional division policy This policy is designed to balance the loads on the links in the network over the long

term by allocating, for each new session, fewer resources on those links that are more highly utilized and thus avoiding the formation of bottleneck links.

Let U_l be the utilization of link $l \in \mathcal{L}(P)$ (fraction of resources that have been allocated), $U_l^+ = \max(U_l, \epsilon)$ ($0 < \epsilon \ll 1$, e.g. $\epsilon = 0.0001$) and Q be the end-to-end QoS to be guaranteed on P. Then the QoS assigned to link $l \in \mathcal{L}(P)$ will be:

$$Q_l = U_l^+ \frac{Q}{\sum_{m \in \mathcal{L}(P)} U_m^+} \quad \forall l \in \mathcal{L}(P) . \qquad (3)$$

A consequence of this allocation is that higher utilized links will be assigned less stringent QoS requirements (higher probability of packet loss or higher delay). This in turn will result in the reservation of fewer resources on that link. Note that U_l^+ has been defined so as to avoid a possible division by zero. COMPUTE_QOS_PATH$(P, V_P, Q; (Q_l)_{l \in \mathcal{L}(P)})$ has the path attribute $V_P = (U_m^+)_{m \in \mathcal{L}(P)}$, computing Q_l as in (3). COMPUTE_QOS_LINK$(P, l, V_P, V_l, Q; Q_l, Q_P)$ has the path attribute $V_P = \sum_{m \in \mathcal{L}(P)} U_m^+$ and the link attribute $V_l = U_l^+$. Q_P is set equal to Q_l, which is computed as in (3).

For all the above QoS division policies, observe that, given a path (A, C) and $B \in \mathcal{N}(A, C)$ we have that $V_{A,B} + V_{B,C} = V_{A,C}$, which is used in the algorithms in Section 3.4.

Division policies for QoS classes In the previous QoS division policies we have assumed that a QoS guarantee can take an arbitrary value within an interval on the real line. There are systems (as in the example in Section 4) where the QoS guarantee can only take one of a finite set of values. Consider the case where Q_l can take the values $q_{1,l} > \cdots > q_{L,l} > 0$ at link l, each value corresponding to a QoS class $j = 1, .., L$. The Even and Proportional division policies can easily be modified to accommodate a finite number of QoS values by truncating each QoS result to the closest smaller value in the set (and thus the end-to-end QoS is still guaranteed). In this case, if Q_l is the value obtained by a division policy and if $q_{i,l} < Q_l < q_{i+1,l}$ for some i then $q_{i,l}$ is the QoS actually allocated on l.

3.2.3 The *uniformity* property

The Even and Proportional division policies exhibit a *uniformity* property that enables us to design an efficient resource reclaim algorithm in Section 3.4.2. The formal definition of the *uniformity* property follows:

Definition 1 *Let $\mathcal{P}$ be a QoS division policy, $P^i = (S, D^i), i = 1, 2$ be two paths sharing a common part $P^1 \cap P^2$, $Q(S, D^i), i = 1, 2$ their end-to-end QoS requirements and $(Q_l^i)_{l \in \mathcal{L}(P^i)}, i = 1, 2$ the results of applying the policy $\mathcal{P}$ on P^1 and P^2. $\mathcal{P}$ is said to be* uniform *if:*

$$\begin{aligned} &\textit{either} \quad Q_l^1 \leq Q_l^2 \quad \forall l \in \mathcal{L}(P^1 \cap P^2) \\ &\textit{or} \quad Q_l^1 \geq Q_l^2 \quad \forall l \in \mathcal{L}(P^1 \cap P^2) . \end{aligned} \qquad (4)$$

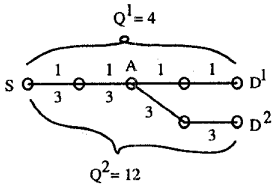

Figure 1: An example of *uniform* QoS division

Uniformity is exemplified in Fig. 1 where the local QoS requirement (=1) from the path (S, D^1) is less than that from path (S, D^2) (=3) for each shared link (i.e. links on (S, A)). We prove the following in [11]:

Theorem 1 *The Even and Proportional division policies are* uniform *division policies.*

The Even and Proportional QoS division policies also exhibit the *uniformity* property when there are a finite number of QoS classes.

The above *uniformity* property allows us to introduce an ordering among paths.

Definition 2 *In the context of a* uniform *division policy, P^1 is* less *than P^2, denoted $P^1 \prec P^2$ if*

$$\exists l \in \mathcal{L}(P^1 \cap P^2) \quad s.t. \quad Q_l^1 < Q_l^2$$

Given a set of paths, the path in the set that is minimum with respect to the relation $\prec$ yields the most stringent QoS requirement on their common part. We refer to this path as *critical* and we use this concept in Section 3.4.2. Given the set of paths $(P^D)_{D \in \mathcal{D}_l}$ that share link l, if either Even or Proportional QoS division is performed on each of them, then the critical path is the one having the minimum QoS path characteristic Q_{PD} derived by COMPUTE_QOS_LINK.

3.3 The allocation phase: admission control and resource reservation

In the static multicast problem, an entire multicast session $\mathcal{M}$ is presented to the network to be established.

The allocation phase is responsible for determining whether there are sufficient resources at the links of the multicast tree $\mathcal{T}$ such that the multicast session $\mathcal{M}$ can be admitted. If $\mathcal{M}$ is admitted, a sufficient amount of resources is allocated at the links in $\mathcal{T}$.

Here we outline the allocation algorithm; centralized and distributed versions are described in detail in [11]. This algorithm is an extension, to multicast sessions, of a scheme proposed for unicast sessions in [10]. It starts by reserving enough resources to accommodate the tightest QoS achievable at each link of the multicast tree $\mathcal{T}$ (essentially peak rate) and checks if the end-to-end QoS thus obtained is no worse than the required one, for each receiver $D \in \mathcal{D}$. For the receivers whose QoS requirements cannot be satisfied, the corresponding resources are released. For each satisfiable receiver, its QoS requirement is mapped into QoS requirements for each link on the path. Finally, a link shared by multiple sender-receiver paths is assigned the tightest (minimum) local QoS requirement. This algorithm has a transactional form, is deadlock free and the transient over-reservation of resources does not significantly impact the network (see [11] for details). Observe that, after assigning the minimum of QoS on a shared link, at least one sender-receiver path containing that link has more resources allocated than necessary to guarantee its end-to-end QoS requirement. The reclaim of such resources in excess is the objective of the reclaim phase.

3.4 The reclaim phase: releasing resources allocated in excess

Once the allocation phase is successfully completed (the multicast session is accepted) the source can start transmitting user data. At the same time, the resources allocated in excess on the multicast tree can be reclaimed without interfering with the user traffic.

3.4.1 The general algorithm

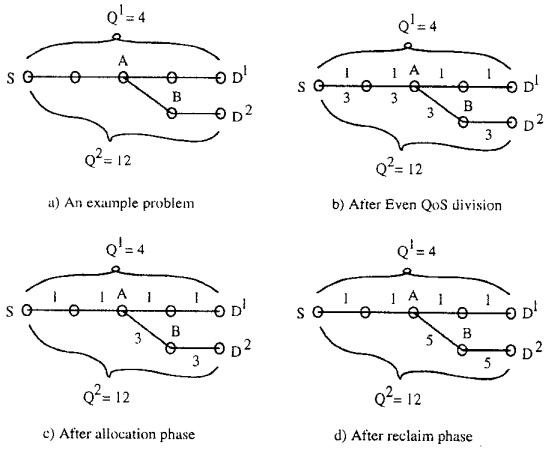

Figure 2: An example of resource allocation and reclaim

The opportunity for resource reclaim phase is exemplified in Fig. 2 where paths (S, D^1) and (S, D^2) partially overlap (2.a). After an Even division of end-to-end QoS requirements (2.b) and assigning the tightest (minimum) QoS to links on their common part (2.c), we have $Q(S, D^2) = 1 + 1 + 3 + 3 = 8 < Q^2$. We can reclaim some resources on (A, D^2): $Q(A, B) = Q(B, D^2) = (12 - 2)/2 = 5$ (2.d).

GENERAL_RECLAIMER(input: $A, \mathcal{D}_A, V_{S,A}, Q(S, A), (V_{S,D})_{D \in \mathcal{D}}$, $(V_m)_{m \in \mathcal{T}}, (Q(S, D))_{D \in \mathcal{D}}$; output: $(Q_l, T_l)_{l \in \mathcal{T}_A}$)

1 **for** l outgoing link from A **do**
2 **if** l is a leaf link in $\mathcal{T}$, $A \xrightarrow{l} D$
3 **then** $Q_l \leftarrow Q(S, D) - Q(S, A)$
4 $T_l \leftarrow F(Q_l)$
5 **else for** $D \in \mathcal{D}_l$ **do**
6 $V_{A,D} \leftarrow V_{S,D} - V_{S,A}$
7 COMPUTE_QOS_LINK($(A, D), l, V_{A,D}, V_l$, $Q(S, D) - Q(S, A); Q_l^D, Q_{(A,D)}$)
8 $Q_l \leftarrow \min_{D \in \mathcal{D}_l} Q_l^D$
9 $T_l \leftarrow F(Q_l)$
10 **let** B be s.t. $A \xrightarrow{l} B$
11 $Q(S, B) \leftarrow Q(S, A) + Q_l$
12 $V_{S,B} \leftarrow V_{S,A} + V_{A,B}$
13 GENERAL_RECLAIMER($B, \mathcal{D}_B, V_{S,B}, Q(S, B)$, $(V_{S,D})_{D \in \mathcal{D}}, (V_m)_{m \in \mathcal{T}}, (Q(S, D))_{D \in \mathcal{D}}; (Q_l, T_l)_{l \in \mathcal{T}_B}$)

Figure 3: The general reclaim algorithm

The reclaim algorithm is given in Fig. 3. The algorithm takes as input a multicast tree rooted at A (denoted by $\mathcal{T}_A$) and the associated end-to-end QoS constraints, and outputs the minimal resource allocation $(T_l)_l$ needed to guarantee the given QoS requirements. For each link l outgoing from A in $\mathcal{T}$, GENERAL_RECLAIMER first allocates resources to guarantee the tightest QoS requirement at link l (lines 5-9) and then calls itself recursively for l's destination node (lines 10-13). To reclaim the resources allocated in excess for the entire tree $\mathcal{T}$, the procedure call is:

$$\text{GENERAL\_RECLAIMER}(S, \mathcal{D}, V_{S,S}, Q(S, S), (V_{S,D})_{D \in \mathcal{D}}, (V_m)_{m \in \mathcal{T}}, (Q(S, D))_{D \in \mathcal{D}}; (Q_l)_{l \in \mathcal{T}}, (T_l)_{l \in \mathcal{T}}) \quad ,$$

where the values for $V_{S,D}, D \in \mathcal{D}$ are known from the allocation phase and $V_{S,S}$ and $Q(S, S)$ are both zero. Taking $N = |\mathcal{D}|$ as the number of destinations and $M = |\mathcal{T}|$ as the number of links in the multicast tree, we show in [11] that GENERAL_RECLAIMER has a worst case running time of $O(NM)$.

3.4.2 An improved algorithm

The complexity of GENERAL_RECLAIMER can be reduced if the QoS division policy is *uniform*. Defining a critical path in $\mathcal{T}$ being the set of links in $\mathcal{T}$ that get their tightest QoS requirement from the same ("critical") receiver, we show in [11] that if the QoS division policy

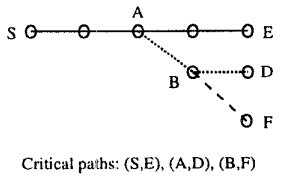

Figure 4: A tree partitioned into critical paths

is *uniform* then $\mathcal{T}$ is partitioned in critical paths (see for an example Fig. 4). The complexity is reduced in IMPROVED_RECLAIMER by computing the resource allocation once per critical path rather than once per link as in GENERAL_RECLAIMER.

IMPROVED_RECLAIMER(input: $S, \mathcal{D}, (V_{S,D})_{D\in\mathcal{D}}, (V_m)_{m\in\mathcal{T}}$, $(Q(S,D))_{D\in\mathcal{D}}$; output: $(Q_l, T_l)_{l\in\mathcal{T}}$)

1 **while** $\exists$ unmarked outgoing link from S (**let** $C = S$)
  or $\exists$ a branching node C with marked incoming link and unmarked outgoing link(s) **do**
2 **for** l unmarked outgoing link from C **do**
3 **for** $D \in \mathcal{D}_l$ **do**
4 $V_{C,D} \leftarrow V_{S,D} - V_{S,C}$
5 COMPUTE_QOS_LINK$((C,D), l, V_{C,D}, V_l, Q(S,D) - Q(S,C); Q_l^D, Q_{(C,D)})$
6 **let** E s.t. $Q_{(C,E)} = \min_{D\in\mathcal{D}_l} Q_{(C,D)}$
7 COMPUTE_QOS_PATH$((C,E), (V_m)_{m\in\mathcal{L}(C,E)}, Q(S,E) - Q(S,C); (Q_m^E)_{m\in\mathcal{L}(C,E)})$
8 **for** $n \in \mathcal{L}(C,E)$ **do**
9 $T_n \leftarrow F(Q_n^E)$
10 **let** A, B s.t. $A \xrightarrow{n} B$
11 $Q(S,B) \leftarrow Q(S,A) + Q_n^E$
12 $V_{S,B} \leftarrow V_{S,A} + V_{A,B}$
13 $mark(n)$

Figure 5: The improved reclaim algorithm

The improved reclaim algorithm given in Fig. 5 starts with the critical paths containing S, divides the QoS requirement and allocates resources for all of its links and marks them as processed. The remaining unmarked forest is then considered, another critical path that starts with a node from a processed path is selected and the procedure is repeated until all of $\mathcal{T}$ is processed.

We prove in [11] that IMPROVED_RECLAIMER reduces the worst case running time to $O(N^2 + M)$ from $O(NM)$ for GENERAL_RECLAIMER. This reduction is important since, in general, the number of leaves (N) of a tree is significantly less than the number of links (M) in that tree.

4 A complete example

In this section we will illustrate the behavior of the various mechanisms with an application. We then evaluate by simulation the effectiveness of resource allocation under the Even and Proportional division policies with and without resource allocation. The network consists of nodes that use Generalized Processor Sharing (GPS) as the link scheduling policy among L classes of sessions (the same set of classes is used for all links in the network). Class k is characterized by the packet loss probability q_k that is guaranteed at each node for all sessions admitted in that class. All packets from sessions within a class at a link are serviced using FIFO scheduling using a common buffer with capacity B_k.

We take as the source traffic for multicast sessions, packetized voice modeled as i.i.d. on/off Markov fluids [4] sending data with a constant rate r during the "on" period and not sending anything during the "off" period. The durations of the "on" and "off" periods are exponential random variables with means $1/\mu$ and $1/\lambda$ respectively. We show in [11] based on [13, 5] that in order to guarantee the loss probability q_k at a link, it is sufficient to reserve the following effective bandwidth at that link:

$$F(q_k) = \frac{\frac{-\log q_k}{B_k} r - \mu - \lambda + \sqrt{(\frac{-\log q_k}{B_k} r - \mu + \lambda)^2 + 4\lambda\mu}}{2\frac{-\log q_k}{B_k}} \tag{5}$$

Given a link with available capacity a and a new session that requires the loss probability q_k at that link, the session can be admitted at that link if $F(q_k) \leq a$. After the resource reservation, the new available capacity of the link becomes $a - F(q_k)$.

4.1 Simulations

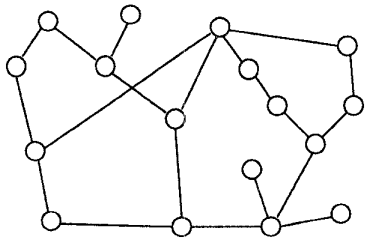

Figure 6: The T3 MBone

The simulations are performed in the context of an existing multicast network topology: the Internet multicast backbone (MBone) as of January 1995, restricted to its T3 (45Mb/s) bidirectional links (Fig. 6). We assume that only 5% of the T3 capacity is used for voice multicast applications, so each unidirectional link will have a bandwidth

of $c = 2.25Mb/s$ allocated for multicast sessions. Multicast sessions are generated according to a Poisson process with parameter α and their durations are exponentially distributed with mean $1/\beta$. $\rho = \alpha/\beta$ characterizes the load offered to the network, i.e. the average number of multicast sessions that would exist at any time in an infinite resource network. Each multicast session has its source and destinations chosen with equal probability from the nodes of the network. The number of destinations is uniformly distributed in the range $1, .., 16$. The sessions' arrival process is an on/off Markov fluid modelling packetized voice having: mean "on" time $1/\mu = 0.352s$, mean "off" time $1/\lambda = 0.650s$, and peak rate $r = 32Kb/s$, values often used in the literature [4]. Each class has a buffer capacity of $B = 30Kb$. The end-to-end loss probability for each source-destination pair is 10^{-a} where the exponent a is uniformly distributed in the range $[1, 3]$. Each node has 10 QoS classes corresponding to the bandwidths $17, 18.\ldots, 26$Kb/s that covers the range of loss probability of $[10^{-7}, 10^{-1}]$. We can see that in this case the relative over-reservation of bandwidth due to admission in the next QoS class is less than 6%. We use the method of independent replications and the confidence intervals in Fig. 7 have 90% confidence level. The same series of multicast calls are simulated under four scenarios: Even division policy (Section 3.2.2) and Proportional division policy (Section 3.2.2) with and without the reclaim phase (Section 3.4). Fig. 7 shows the network's call rejection probability as a function of the offered load ρ, and Fig. 8 shows the relative difference in the performance of reclaimed Even, non-reclaimed Proportional and reclaimed Proportional with respect to non-reclaimed Even as a function of offered load. We can see that both the reclaimed Even division and non-reclaimed Proportional division bring significant gain (lower blocking probability) compared to the non-reclaimed Even division. This gain is further increased if we combine the two in the reclaimed Proportional division, as seen in Fig. 8 that plots the relative gain.

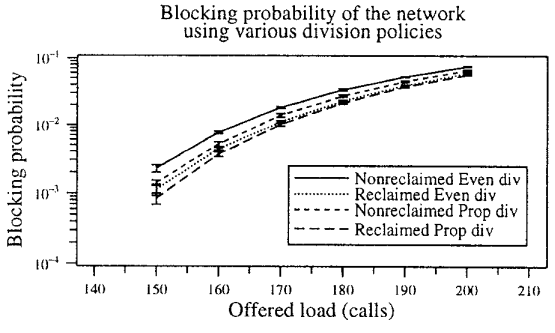

Figure 7: The blocking probability v.s. offered load

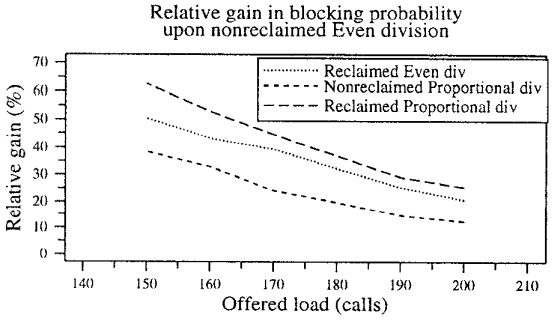

Figure 8: The relative gain

5 Discussions, conclusions and future work

We have developed solutions to the problems of call admission control and resource reservation for a multicast application once a route (multicast tree) has been chosen. The algorithm consists of two phases. The first is responsible for determining whether or not the call can be accepted and, if so, reserving sufficient resources to guarantee the QoS requirements. As the resource allocation is based on considering the multicast session as a set of unicast sessions, the reclaim phase is responsible for refining the allocation by accounting for the fact that different destinations which share a path segment may differ in their local QoS requirements on this segment. A set of efficient reclaim algorithms based on the *uniformity* property exhibited by the Even and Proportional QoS division policies was presented. A preliminary evaluation of these algorithms was presented in a stochastic GPS network, where it was observed that the reclaimed Proportional QoS division policy provides a 20% to 60% decrease in call rejection probability compared to other policies.

Although we used packet loss probability as the QoS metric in our example, all of the algorithms apply equally as well in the case that the QoS metric is maximum packet delay. Furthermore, the algorithms can be applied to QoS metrics such as the probability that the end-to-end delay exceeds a known quantity and a bound on delay jitter with minor modifications. Our allocation and reclaim algorithms can be extended to be receiver oriented (where receivers initiate their join/leave to the multicast session) and to be distributed (where the computations and necessary data for allocation and reclaim of resources are distributed in the nodes of the multicast tree). This is the subject of our ongoing research. Also open for future investigation is the analysis of new QoS division policies. Even though the Proportional division policy has proven better than the Even division, we believe that some nonlinear dependencies of resource allocation with the link utilization would result in an even better network blocking probability. An-

other interesting direction of study is the behavior of various QoS division policies in the context of session correlations as is the case of filters proposed by RSVP.

Acknowlegdements

The authors would like to thank Jim Kurose and Zhi-Li Zhang for many useful discussions.

References

[1] D. A. Anderson. Metascheduling for Continuous Media. *ACM Transactions on Computer Systems*, 11(3), August 1993.

[2] T. Ballardie, P. Francis, and J. Crowcroft. Core Based Trees. In *ACM SIGCOMM '93*, pages 85–95, September 1993.

[3] A. Birman, V. Firoiu, R. Guérin, and D. Kandlur. Provisioning of RSVP-based Services over a Large ATM Network. Technical Report RC 20250, IBM T.J. Watson Research Center, November 1995. http://www.research.ibm.com:8080/main-cgi-bin /search_paper.pl/entry_ids=7844.

[4] P. T. Brady. A statistical analysis of on-off patterns in 16 conversations. *Bell System Technical Journal*, 47:73–91, January 1968.

[5] G. de Veciana, C. Courcoubetis, and J. Walrand. Decoupling Bandwidths for Networks. Technical Report M93/50, UCB/ERL, June 1993.

[6] S. E. Deering and D. R. Cheriton. Multicast Routing in Datagram Internetworks and Extended LANs. *ACM Transactions on Computer Systems*, 8(2):85–110, May 1990.

[7] M. Doar and I. Leslie. How Bad is Naïve Multicast Routing ? In *IEEE INFOCOM '93*, pages 82–89, 1993.

[8] W. Effelsberg and E. Müller-Menrad. Dynamic Join and Leave for Real-Time Multicast. Technical Report TR-93-056, Tenet Group, U.C. Berkeley and ICSI, October 1993.

[9] A. I. Elwalid and D. Mitra. Effective Bandwidth of General Markovian Traffic Sources and Admission Control of High Speed Networks. *Transactions on Networking*, 1(3), June 1993.

[10] D. Ferrari and D. Verma. A Scheme for Real-Time Channel Establishment in Wide-Area Networks. *IEEE Journal on Selected Areas in Communications*, 8(3), April 1990.

[11] V. Firoiu and D. Towsley. Call Admission and Resource Reservation for Multicast Sessions. Technical Report TR 95-17, Univ. of Massachusetts, Amherst, 1995. ftp://ftp.cs.umass.edu/pub/techrept/techreport /1995/UM-CS-1995-017.ps.

[12] R. Frederik. *nv*. Manual Pages, Xerox Palo Alto Research Center.

[13] R. Guérin, H. Ahmadi, and M. Naghshineh. Equivalent Capacity and Its Application to Bandwidth Allocation in High-Speed Networks. *IEEE Journal on Selected Areas in Communications*, 9:968–981, September 1991.

[14] V. Jacobson and S. McCanne. *vat*. Manual Pages, Lawrence Berkeley Laboratory, Berkeley, CA.

[15] C. A. Noronha Jr. and F. A. Tobagi. Optimum Routing of Multicast Streams in Communications Networks. Technical Report CSL-TR-94-618, Department of Electrical Engineering and Computer Science, Stanford University, April 1994.

[16] A. K. Parekh. *A Generalized Processor Sharing Approach to Flow Control in Integrated Services Networks*. PhD thesis, Department of Electrical Engineering and Computer Science, MIT, February 1992.

[17] H. Schulzrinne. Voice Communication Accross the Internet: A Network Voice Terminal. Technical Report TR-92-50, Department of Computer Science, University of Massachusetts, Amherst, July 1992.

[18] C. Topolcic. *Experimental Internet Stream Protocol: Version 2 (ST-II)*. Internet RFC 1190, October 1990.

[19] B. M. Waxman. Performance Evaluation of Multipoint Routing Algorithms. In *IEEE INFOCOM '93*, pages 980–986, 1993.

[20] O. Yaron and M. Sidi. Calculating Performance Bounds in Communication Networks. In *IEEE INFOCOM '93*, 1993.

[21] L. Zhang, S. E. Deering, D. Estrin, S. Shenker, and D. Zappala. RSVP: A New Resource ReSerVation Protocol. *IEEE Network*, 7(5):8–18, September 1993.