Call Admission and Resource Reservation for Multicast Sessions *

Victor Firoiu
vfiroiu@cs.umass.edu

Don Towsley
towsley@cs.umass.edu

Computer Science Department
Lederle Graduate Research Center
University of Massachusetts
Amherst, MA 01003-4610 USA

CMPSCI Technical Report TR 95-17

September 1995

Abstract

Multicast applications, including audio and video, are expected to consume a large fraction of resources in forthcoming high speed networks. Because of this, new services are needed to provide the quality of service (QoS) required by these applications. In this paper we take a step in this direction by presenting 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 allocation, and the optimization of the resulting resource allocation for a multicast session. 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. A comprehensive application of an instance of the algorithms in the context of packetized voice multicast connections over the Mbone is provided to illustrate their applicability.

KEYWORDS: Multicast Algorithms, Quality of Service, Resource Allocation and Optimization, Admission Control.

*This material is based upon work supported 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.

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 applications such as audio (vat [JM], NeVoT [Sch92]), video (nv [Fre]) 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 additive quality of service (QoS) guarantees such as 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 central result of our work is a set of algorithms which take the resource allocation produced by such an approach and optimizes it by reclaiming 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 in networks, with applications to call admission of unicast connections (see[dVCW93, GG92, Kel91, KWC93, Par92, YS93]). The allocation of resources has been another topic of research in recent years and various protocols have been developed: RSVP ([ZDE+93]), ST-II ([Top90]), (also see [FV90, MESZ94, And93]). 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 [DC90, BFC93, DL93, EMM93, NT94, Wax88, Wax93]). 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 route establishment and call admission of multicast sessions, [NT94, Wax88, Wax93]. 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 [Wax93] 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 [DL93].

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 optimization 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) and then extended to solve a dynamic multicast problem (where receivers can join or leave anytime during the life of the session).

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

We formalize the problem of call admission and resource allocation for multicast sessions as follows. We begin with some notation:

- A network is represented by a directed graph $G(V, E)$ with weights associated to 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 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 solving the above problem we make the following assumptions:

- The multicast tree $\mathcal{T}$ is known.
- The QoS metric considered in this paper is the probability of packet loss due to buffer overflow. This is appropriate for traffic such as voice and video which can tolerate a small fraction of packets being lost in the network. For example $Q(S, D) = 0.01$ means that the session can tolerate a loss of no more than 1% of the packets going from S to D. The solutions developed here can also be applied to other kinds of QoS metrics based on delay, delay jitter, etc. as discussed in Section 5.
- The resource needed by a session on a link in order to have its local QoS requirement guaranteed consists of a fraction of the total bandwidth of that link. All links are considered to be simplex (unidirectional).
- The resource requirements increase as local QoS requirements become more stringent. This assumption is natural (in any system better performance requires more resources) and holds in the application presented in Section 4.
- Each node (router) includes a switch which is responsible for packet forwarding and scheduling on all outgoing links and a control processor which is responsible for resource bookkeeping and admission control.
- No costs of any kind are considered. We assume that costs are accounted for during the routing phase and, because we assume that the choice of route is performed prior to admission control, the costs can be considered as embedded in the routing decisions that have already been made.

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 or not 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 $\mathcal{M}$ can be admitted, the allocation phase performs an initial allocation of resources at the links within $\mathcal{T}$ to satisfy those requirements. A second optimization phase is concerned with optimizing the resource allocation in $\mathcal{T}$ by taking advantage of situations where different destinations share a path segment and impose different resource requirements on that segment.

The allocation phase algorithm is based on the following functionally independent components:

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

The allocation phase algorithm is described in Section 3.3 and the optimization phase algorithm in Section 3.4 for the static multicast problem. Section 3.5 describes algorithms for allocation and optimization phases of the dynamic multicast problem.

3.1 The mechanism to relate the local QoS requirement to the resource to be allocated at the link

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) [Cru91, Par92], exponentially bounded burstiness (EBB) [YS93, ZTK94], on/off Markov fluid [GH91, GAN91], etc.) and the link scheduling policy (First Come First Served (FCFS), Generalized Processor Sharing (GPS) [Par92, ZTK94], Earliest Deadline First (EDF) [FV90, GGP94], etc.). It is possible that no explicit mapping between local QoS requirement and resources exists in the case of some link scheduling policies. In such cases procedures are required to map the end-to-end QoS requirements directly to the local resources needed at the links.

In the case that it is possible to map local QoS requirements to resources, we assume that the link scheduling policy exhibits the following two properties. First, the link scheduling policy should support sessions requiring different local QoS guarantees at a link. Not all policies permit this. For instance, per session QoS requirements cannot be guaranteed by a FIFO link scheduling policy. However, policies such as Generalized Processor Sharing (GPS) do permit this (see for example [Par92]).

A second, highly desirable (but not mandatory) property is that a newly accepted session should not affect any of the pre-existing sessions' QoS guarantees at a link. The absence of this property implies the need to recompute the resource allocations of all the affected sessions as a new session is admitted. This is likely to be computationally intractable. This happens for example in the case of deterministic worst-case GPS (as analyzed in [Par92]). Fortunately, there are policies for which this property holds, including a stochastic GPS link scheduling policy that will be part of our example in Section 4.

Henceforth, we assume the existence of the following functions:

$$F_l(Q) \mapsto T$$

where Q is the local QoS requirement at link l and T is the amount of resources needed at l to guarantee Q; and

$$H_l(T) \mapsto Q$$

such that H_l is the inverse function of F_l ($H_l = F_l^{-1}$). Such a function can be determined for a given combination of QoS requirement, session arrival process characteristics, and link scheduling policy via performance analyses (for example [KWC93, GAN91]).

3.2 The mapping of 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 packet 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. The QoS division is implemented in the following procedure:

$$\text{COMPUTE\_QOS\_PATH}(P, (V_m)_{m\in\mathcal{L}(P)}, Q; (Q_m)_{m\in\mathcal{L}(P)}) \tag{1}$$

which takes as input a path $P = (A, D)$ in $\mathcal{T}$, an attribute V_m for each link $m \in \mathcal{L}(P)$, and the end-to-end QoS requirement Q for P. It outputs the local QoS requirements Q_m for each $m \in \mathcal{L}(P)$ such that $\sum_{m\in\mathcal{L}(P)} Q_m \leq Q$ and $F(Q_m) \leq R_m \quad \forall m \in \mathcal{L}(P)$ where $(R_m)_{m\in\mathcal{L}(P)}$ are the local available resources. 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, a link l, a path attribute V_P, a link attribute V_l, and the QoS requirement Q for P as input, then the procedure:

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

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 will be described in Section 3.2.2. There are several issues related to this division of end-to-end QoS such as how to compose the local QoS requirements to reconstruct the end-to-end QoS requirement and how to distribute the QoS requirement to links. These issues are considered in the following subsections.

3.2.1 The calculus of QoS division

Let $A \xrightarrow{l} B \xrightarrow{m} C$ be two adjacent links in the network G. An upper bound for the end-to-end loss probability is:

$$\begin{aligned}
\Pr(\text{loss on } (A,C)) &= \Pr((\text{loss on } (A,B)) \vee (\text{loss on } (B,C))) \\
&= \Pr(\text{loss on } (A,B)) + \Pr(\text{loss on } (B,C)) - \\
&\quad \Pr((\text{loss on } (A,B)) \wedge (\text{loss on } (B,C))) \\
&< \Pr(\text{loss on } (A,B)) + \Pr(\text{loss on } (B,C))
\end{aligned}$$

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

$$Q(A,C) \approx Q_l + Q_m \ .$$

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 attempts to divide the QoS requirements evenly among the links, whereas the

second allocates more stringent QoS requirements to links having more available resources. We end the section with a discussion on how to apply the above policies in the case where QoS guarantees can take only a finite set of values.

Even division policy This policy implements the simple heuristic of allocating 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$(P, (V_m)_{m\in\mathcal{L}(P)}, Q; (Q_m)_{m\in\mathcal{L}(P)})$, the end-to-end QoS requirement Q on path P is divided into $|\mathcal{L}(P)|$ equal parts: $Q_l = Q/\ |\mathcal{L}(P)|, \quad \forall l \in \mathcal{L}(P)$. As this computation requires no other information regarding the path, the link attributes $V_m = \emptyset\ \forall m \in \mathcal{L}(P)$. The computational complexity is thus $O(|\mathcal{L}(P)|)$ since the computation of $|\ \mathcal{L}(P)\ |$ and the assignment of values to $(Q_l)_{l\in\mathcal{L}(P)}$ are each $O(|\ \mathcal{L}(P)\ |)$. In COMPUTE_QOS_LINK$(P, l, V_P, V_l, Q; Q_l, Q_P)$ the path attribute is $V_P = |\ \mathcal{L}(P)\ |$ and the outputs are $Q_l = Q_P = Q/\ |\mathcal{L}(P)|$ for a specified $l \in \mathcal{L}(P)$. As this computation requires no other information regarding the link l, the link attribute $V_l = \emptyset$. Consequently, it has a computational complexity of $O(1)$.

Consider a path (A, C) and an intermediate node $B \in \mathcal{N}(A, C)$. Form our definition of the path attribute V_P it is easy to derive the following operations:

$$V_{A,B} \oplus V_{B,C} \triangleq V_{A,C} = |\ \mathcal{L}(A,B)\ | + |\ \mathcal{L}(B,C)\ | \quad ,$$

$$V_{A,C} \ominus V_{A,B} \triangleq V_{B,C} = |\ \mathcal{L}(A,C)\ | - |\ \mathcal{L}(A,B)\ | \quad .$$

The above operations are used in the algorithms in Section 3.4. Observe that both the above operations have a computational complexity of $O(1)$.

We consider now the case where one or more links has an insufficient amount of resources to accommodate an even division of QoS. As we will soon see, COMPUTE_QOS_PATH and COMPUTE_QOS_LINK are invoked only if a path has sufficient resources to accept a session. We consider first COMPUTE_QOS_PATH$(P, (V_m)_{m\in\mathcal{L}(P)}, Q; (Q_m)_{m\in\mathcal{L}(P)})$ with the path P, the most stringent QoS requirement that each link m can support $V_m = Q_m^{min}\ m \in \mathcal{L}(P)$ and the QoS requirement Q as inputs, and with $(Q_m)_{m\in\mathcal{L}(P)}$, the local QoS requirements as output. We have

$$Q \geq \sum_{m\in\mathcal{L}(P)} Q_m^{min}\ .$$

A problem arises if $Q/\ |\mathcal{L}(P)| < Q_l^{min}$ for some $l \in \mathcal{L}(P)$. The even division policy is modified so that a link that is resource limited allocates all of its available resources to the session, i.e. assign $Q_l \leftarrow Q_l^{min}$ at that link. The rest of the end-to-end QoS is then evenly assigned among the remaining links on the path. If this is not possible, the procedure is repeated however many times as is required. Formally, given the end-to-end QoS Q to be divided and $(Q_l^{min})_{l\in\mathcal{L}(P)}$ the set of minimum local QoS, the assigned local QoS are:

$$Q_l = \max(Q_l^{min}, Q_{free}) \tag{3}$$

such that $\sum_{l \in \mathcal{L}(P)} Q_l = Q$ and Q_{free} is the QoS allocation given to the links that are found not to be resource limited.

COMPUTE_QOS_PATH(input: $P, (Q_m^{min})_{m \in \mathcal{L}(P)}, Q$; output: $(Q_m)_{m \in \mathcal{L}(P)}$)

```
1  Q_rest ← Q
2  N_rest ←|L(P)|
3  for l ∈ L(P) in descending order of Q_l^min do
4      if Q_l^min ≤ Q_rest/N_rest
5        then Q_free ← Q_rest/N_rest
6             for m ∈ L(P) in descending order of Q_m^min starting with l do
7                 Q_m ← Q_free
8             return
9        else
10            Q_l ← Q_l^min
11            Q_rest ← Q_rest − Q_l^min
12            N_rest ← N_rest − 1
```

Figure 1: Even QoS division with resource limitations

The algorithm for computing Q_{free} is shown in Figure 1. The set $(Q_l^{min})_{l \in \mathcal{L}(P)}$ is sorted and the list of links in $\mathcal{L}(P)$ is processed in descending order. A check is made to see if, by dividing the end-to-end QoS evenly among all links, the resulting share is larger than the largest Q^{min}. If so, all the links can accommodate this share and the algorithm terminates. If not, assign to the current link in the list its Q^{min}, subtract it from the QoS to be divided, advance to the next link in the list and repeat the step for the remaining links. Eventually the algorithm will be able to divide the rest of the QoS evenly among the rest of the links. We shall refer to this equal share as Q_{free}.

The procedure COMPUTE_QOS_LINK$(P, l, V_P, V_l, Q; Q_l, Q_P)$ computes the required QoS Q_l for a specified link l and outputs as a path characteristic $Q_P = Q_{free}^P$. The inputs are: a path P, a link l, the most stringent QoS requirement that each link m can support $V_P = (Q_m^{min})_{m \in \mathcal{L}(P)}$ and the QoS requirement Q. As this computation requires no other information regarding the link l, the link attribute $V_l = \emptyset$. It is the same algorithm as in Figure 1 with an appropriate change in the input/output parameters. To analyze the computational complexity of both COMPUTE_QOS_PATH and COMPUTE_QOS_LINK, we observe that the number of loops in Figure 1 is $O(|\mathcal{L}(P)|)$ and that there is an implicit sort of $O(|\mathcal{L}(P)| \log |\mathcal{L}(P)|)$, giving a total of $O(|\mathcal{L}(P)| \log |\mathcal{L}(P)|)$.

Consider a path (A, C) and an intermediate node $B \in \mathcal{N}(A, C)$. Form our definition of the path attribute V_P it is easy to derive the following operations:

$$V_{A,B} \oplus V_{B,C} \stackrel{\triangle}{=} V_{A,C} = \text{augment the set } (Q_m^{min})_{m \in \mathcal{L}(A,B)} \text{ with the set } (Q_m^{min})_{m \in \mathcal{L}(B,C)} \quad ,$$

$$V_{A,C} \ominus V_{A,B} \stackrel{\triangle}{=} V_{B,C} = \text{restrict the set } (Q_m^{min})_{m \in \mathcal{L}(A,C)} \text{ to the set } (Q_m^{min})_{m \in \mathcal{L}(B,C)} \quad .$$

The above operations are used in the algorithms in Section 3.4. Observe that both the above operations have a computational complexity of $O(1)$ if we implement the sets as linked lists.

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:

$$Q_l = U_l^+ \frac{Q}{\sum_{m \in \mathcal{L}(P)} U_m^+} \quad \forall l \in \mathcal{L}(P) \; . \tag{4}$$

A consequence of this allocation is that higher utilized links will be assigned less stringent QoS requirements (higher probability of packet loss). This in turn will result in the reservation of fewer resources on that link. Note that U_l^+ has been defined so to avoid a possible division by zero. The procedure COMPUTE_QOS_PATH$(P, (V_m)_{m \in \mathcal{L}(P)}, Q; (Q_m)_{m \in \mathcal{L}(P)})$ computes the local QoS requirements $(Q_m)_{m \in \mathcal{L}(P)}$ as in (4) having as inputs the QoS requirement Q for P and the link attributes $V_m = U_m^+ \; \forall m \in \mathcal{L}(P)$. It thus has a computational complexity of $O(|\mathcal{L}(P)|)$ because computing $\sum_{m \in \mathcal{L}(P)} U_m^+$ and the computation of Q_l for each $l \in \mathcal{L}(P)$ are each $O(|\mathcal{L}(P)|)$. The procedure COMPUTE_QOS_LINK$(P, l, V_P, V_l, Q; Q_l, Q_P)$ computes the local QoS requirement Q_l for a specified link l as in (4) having as inputs the QoS requirement Q for P, 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. It thus has a computational complexity of $O(1)$.

Consider a path (A, C) and an intermediate node $B \in \mathcal{N}(A, C)$. Form our definition of the path attribute V_P it is easy to derive the following operations:

$$V_{A,B} \oplus V_{B,C} \stackrel{\triangle}{=} V_{A,C} = \sum_{m \in \mathcal{L}(A,B)} U_m^+ + \sum_{m \in \mathcal{L}(B,C)} U_m^+ \quad ,$$

$$V_{A,C} \ominus V_{A,B} \stackrel{\triangle}{=} V_{B,C} = \sum_{m \in \mathcal{L}(A,C)} U_m^+ - \sum_{m \in \mathcal{L}(A,B)} U_m^+ \quad .$$

The above operations are used in the algorithms in Section 3.4. Observe that both the above operations have a computational complexity of $O(1)$.

Division policies for QoS classes In the previous QoS division policies we have assumed that a QoS guarantee can take a value within an interval on the real line. There are systems (e.g. 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 $j = 1, .., L$ (see Section 4.1 and 4.2 for an example of QoS classes). The Even and Proportional division policies can easily be modified to accommodate a finite set of QoS values by

truncating their QoS results to the closest 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 let

$$g(Q_l) = q_{i,l} \tag{5}$$

denote the QoS actually allocated on l.

In the following we discuss only the specific issues of division with QoS classes. If:

$$Q(S,D) < \sum_{l \in \mathcal{L}(P^D)} q_{L,l} \quad ,$$

then the call adimission test fails and the session is rejected. If, on the other hand:

$$Q(S,D) > \sum_{l \in \mathcal{L}(P^D)} q_{1,l} \quad ,$$

the session is admitted in the highest QoS class (class 1) at all of the links on the path. Between the two extremes, multiple combinations of classes along a path are possible for a given QoS division policy. In the case of multicasting, for a given combination $(Q_l)_{l \in \mathcal{L}(P^D)}$, we prefer to place lower QoS class allocations in positions closer to source S because it is more likely for a link closer to the source to be exposed to a wider range of local QoS requirements from the multiple sessions passing through it, and thus, the minimum of these requirements is more likely to be a lower QoS.

3.2.3 The *uniformity* property

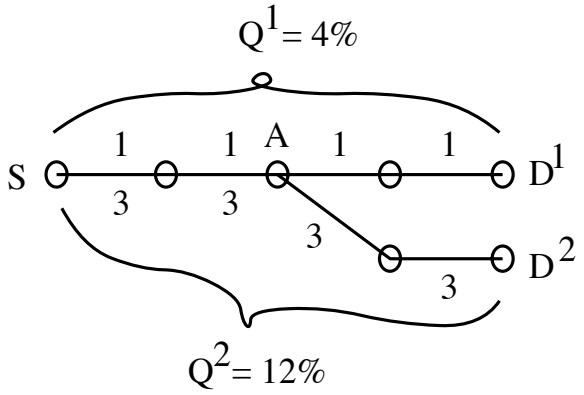


Figure 2: An example of *uniform* QoS division

The QoS division policies presented in the previous section produce "uniform" assignments of QoS requirements to links. Consider two paths to destinations D^1 and D^2 in a multicast tree that share two or more links. An assignment is *uniform* if the QoS requirement for D^1 is either greater than or less than that for D^2 at all links on their common part. This is illustrated in Figure 2 where we observe that for each link on the common path (S, A) the local QoS $(= 1)$ from the first path is

less than the corresponding local QoS (= 3) from the second path. This property will form a basis for an efficient algorithm used in the optimization phase (presented 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:*

$$\textit{either } Q_l^1 \leq Q_l^2 \, \forall l \in \mathcal{L}(P^1 \cap P^2) \quad \textit{or } Q_l^1 \geq Q_l^2 \, \forall l \in \mathcal{L}(P^1 \cap P^2) \, . \tag{6}$$

In the following we assume that P^1 and P^2 are established simultaneously or close enough in time such that the network values that are used for QoS division computation for both P^1 and P^2 are the same. This assumption is valid in the case where the multicast session is established at the same time for all receivers. At the end of this section we examine the case where this assumption does not hold, as in the case where receivers join or leave independently the multicast session. We now establish that the Even and Proportional division policies are uniform division policies.

Theorem 1 *The even and proportional division policies are* uniform *division policies.*

Proof

Consider two paths $P^i = (S, D^i), i = 1, 2$ that share a common part. In the case that there are no resource limitations, the even division policy exhibits the *uniformity* property because we have

$$Q_n^1 = Q_m^1 \quad \forall n, m \in \mathcal{L}(P^1) \quad \text{and} \quad Q_n^2 = Q_m^2 \quad \forall n, m \in \mathcal{L}(P^2)$$

so, if

$$\exists n \in \mathcal{L}(P^1 \cap P^2) \quad \text{such that} \quad Q_n^1 < Q_n^2$$

then

$$Q_m^1 < Q_m^2 \quad \forall m \in \mathcal{L}(P^1 \cap P^2) \quad .$$

The even QoS division with resource limitations exhibits the *uniformity* property as shown in the following. Let $P^1 = (S, D^1)$, $P^2 = (S, D^2)$ be two paths, $Q(S, D^1)$ and $Q(S, D^2)$ their end-to-end QoS requirements and $(Q_n^1)_{n \in \mathcal{L}(P^1)}$, $(Q_n^2)_{n \in \mathcal{L}(P^2)}$ the local QoS requirements obtained by applying the even division policy in the case that one or more of the links on either path are resource limited. We will prove that (6) holds. In the proof we need the following observation that can easily be derived from (3):

Observation 1 *Let P be a path, Q_n^{min} the minimum QoS available in n, Q_n the QoS obtained in link $n \in \mathcal{L}(P)$ after running the QoS division algorithm with resource limitations and Q_{free} the QoS allocation given to the links in $\mathcal{L}(P)$ that are not resource limited. Then either $Q_n = Q_n^{min}$ or $Q_n = Q_{free}$.*

Now we can proceed to prove the Theorem. Consider $n \in \mathcal{L}(P^1 \cap P^2)$ such that

$$Q_n^1 > Q_n^2 \,. \tag{7}$$

We assume also that there exists $m \in \mathcal{L}(P^1 \cap P^2)$, $m \neq n$ such that

$$Q_m^1 < Q_m^2 \,, \tag{8}$$

and proceed to find a contradiction. From (7) we have:

$$Q_n^1 > Q_n^2 \geq Q_n^{min} \,. \tag{9}$$

since $Q_n^1 \neq Q_n^{min}$, we have that $Q_n^1 = Q_{free}^1$ from Observation 1. According to (3), this implies

$$Q_m^1 \geq Q_{free}^1 = Q_n^1 \tag{10}$$

and we consider two cases (see Figure 3).

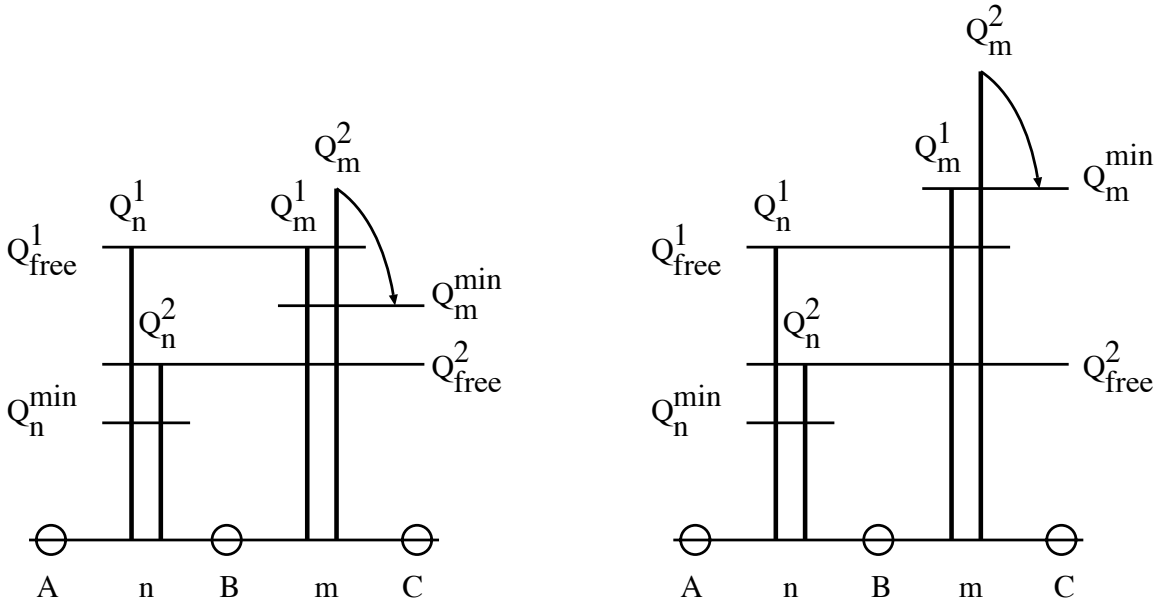


Figure 3: The two cases for proving *uniformity* of even division policy with resource limitations

Case 1 $Q_m^1 = Q_n^1$. Then from (10)

$$Q_m^1 = Q_n^1 = Q_{free}^1 \,, \tag{11}$$

and from (3)

$$Q_m^1 \geq Q_m^{min} \,. \tag{12}$$

But from (8), from the hypothesis of this case and from (7) respectively,

$$Q_m^2 > Q_m^1 = Q_n^1 > Q_n^2 \geq Q_{free}^2 \tag{13}$$

This implies $Q_m^2 > Q_{free}^2$ and from Observation 1 and from (12) we have:

$$Q_m^2 = Q_m^{min} \leq Q_m^1 \tag{14}$$

which contradicts the statement $Q_m^2 > Q_m^1$ in (13).

Case 2 $Q_m^1 > Q_n^1$. From Observation 1

$$Q_m^1 = Q_m^{min} \tag{15}$$

and:

$$Q_m^2 > Q_m^1 > Q_n^1 > Q_n^2 \geq Q_{free}^2 \tag{16}$$

where the first inequality comes from (8), the second from the hypothesis of this case, the third one from (7) and the last from (3). Then, from Observation 1, $Q_m^2 = Q_m^{min}$ and from (15):

$$Q_m^1 = Q_m^2 \tag{17}$$

which contradicts (8).

Because both cases lead to contradictions, the supposition in (8) is refuted and Theorem 1 proven.

Consider now the proportional division policy. From (4) we have

$$Q_n^1/U_n^+ = Q_m^1/U_m^+ \quad \forall n, m \in \mathcal{L}(P^1) \quad \text{and } Q_n^2/U_n^+ = Q_m^2/U_m^+ \quad \forall n, m \in \mathcal{L}(P^2) \quad .$$

so, if

$$\exists n \in \mathcal{L}(P^1 \cap P^2) \quad \text{such that} \quad Q_n^1 < Q_n^2$$

then

$$\frac{U_n^+}{U_m^+} Q_m^1 < \frac{U_n^+}{U_m^+} Q_m^2 \Longrightarrow Q_m^1 < Q_m^2 \quad \forall m \in \mathcal{L}(P^1 \cap P^2)$$

and thus the property in (6) holds. ∎

The even and proportional QoS division policies exhibit the *uniformity* property also when the number of QoS values is finite. The proofs follow the above proofs closely, replacing any Q_n value with its floor defined in (5) and observing that this operation does not change the sense of any inequality.

Suppose now that P^1 and P^2 are not established simultaneously. Then the even QoS division without resource limitations is still *uniform* because $Q_n^1 = Q_m^1$ and $Q_n^2 = Q_m^2$ at any time and the relations in (6) are verified. Even division with resource limitations is no longer *uniform* because the values (Q_n^{min}) can vary in time, and it is possible to have: $Q_{free}^1 < Q_n^{min,2}$ and $Q_{free}^2 < Q_m^{min,1}$. Proportional division is not *uniform* because the values $(U_m^+)_m$ can vary in time and it is possible to have $Q_n^1 < Q_n^2$ and $Q_m^2 < Q_m^1$.

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*

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

This $\prec$ relation can be used to order paths. Given a set of paths, their minimum with respect to the relation $\prec$ yields the most stringent QoS requirement on their common part, is called a critical path and is used in Section 3.4.2. We examine next how this ordering can be established for the proposed QoS division policies.

In the case of the even division policy without resource limitations we can associate with each path sharing a link l the QoS requirement on l obtained from dividing the end-to-end QoS requirements for each path P^D $D \in \mathcal{D}_l$. The paths are thus ordered based on $(Q_l^D)_{D \in \mathcal{D}_l}$.

In the case of even division with resource limitations we can associate with each path sharing a link l the QoS requirement to be assigned to links on that path that do not have resource limitation. The paths are thus ordered based on $(Q_{free}^D)_{D \in \mathcal{D}_l}$.

In the case of proportional division we can associate with each path sharing the link l the QoS requirement on l obtained from proportionally dividing the end-to-end QoS requirements for each path P^D $D \in \mathcal{D}_l$. The paths are thus ordered based on $(Q_l^D)_{D \in \mathcal{D}_l}$.

In all cases, the path characteristic Q_l^D or Q_{free}^D is output by the procedure COMPUTE_QOS_LINK in the parameter Q_{PD}, as presented at the beginning of Section 3.2.

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. We have shown in Section 3.2.3 that in this case both even and proportional division policies are *uniform.* Depending on the admission criterium used, the multicast session is admitted if all or a part of receivers can obtain their QoS requirements (see discussion in Section 5).

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 $\mathcal{T}$.

A centralized version The routed path from the source S to a destination D in the multicast tree $\mathcal{T}$ is $P^D = (S, D)$, $D \in \mathcal{D}$. For each of these paths, using either the amount of resources required to guarantee the most stringent QoS possible on $l \in \mathcal{L}(P^D)$ (peak rate) or the total amount of available resources on l, whichever is smaller (call it R_l'), check if the end-to-end QoS requirement $Q(S, D)$ can be met based on the QoS guarantees that correspond to the resources R_l' ($Q_l = F(R_l')$). Observe that, if link l is shared by two or more paths, the same resource R_l' is used for each path because multicasting allows a single copy of data to be propagated on common parts of paths. Based on the outcome of the above tests and on the grade of success required (described in Section 5) the multicast session is (partially) accepted or rejected. Based on the end-to-end/nodal division policy (described in Section 3.2) and on nodal QoS/resource relation (described in Section 3.1), for

each path P^D and for each link $l \in \mathcal{L}(P^D)$ compute a resource allocation $T_l^D \leq R'_l$ such that all (accepted) destinations meet their end-to-end QoS requirements $Q(S, D)$. Allocate for session $\mathcal{M}$ at each link l the resource $T_l = \max_{D \in \mathcal{D}_l} T_l^D$ (where $\mathcal{D}_l = \{D \in \mathcal{D} \mid l \in \mathcal{L}(P^D)\}$) by subtracting T_l from the available resources R_l. Observe that this assignment will preserve the QoS guarantees already accepted and that for some paths the resource allocations may be greater than needed. For those paths the resources allocated can be reduced using the algorithm described in Section 3.4.

A distributed version The algorithm described here is distributed and uses an idea proposed in [FV90] for the case of unicast sessions and to a certain extent in [EMM93] for multicast sessions.

After the multicast call is requested and the multicast tree is routed, for each destination $D \in \mathcal{D}$ a message containing the QoS requirement $Q(S, D)$ is sent from the source S to D, pre-reserving either the amount of resources required to guarantee the most stringent QoS possible on $l \in \mathcal{L}(P^D)$ (peak rate) or the total amount of available resources on l, whichever is smaller (call it R'_l). The messages are sent in parallel and the resources are pre-reserved only once on common links. The corresponding QoS information ($Q_l = F(R'_l)$) is added to the message. As the message arrives at the destination node D, a check is made on whether to accept the unicast session over the path (S, D) based on the carried information $(Q(S, D), (Q_l)_{l \in \mathcal{L}(P^D)})$. If the unicast session is rejected, a reply is returned to the source and the pre-reserved resources are released in those links that are not shared by other accepted unicast components of $\mathcal{M}$. If the unicast session is accepted, another computation is done at the destination node D to determine the actual resources needed at the links on the path P^D based on the end-to-end/nodal division policy (described in Section 3.2) and on nodal QoS/resource relation (described in Section 3.1). The reply, on its way back to the source, disseminates this information to the links. At each link l, after all the messages for the paths that share l have passed back, the amount $T_l = \max_{D \in \mathcal{D}_l} T_l^D$ is computed ($\mathcal{D}_l = \{D \in \mathcal{D} \mid l \in \mathcal{L}(P^D)\}$) and is allocated to the session (by subtracting it from the available resources, R_l). At the source, after all messages have returned, a decision whether to accept or reject the entire multicast session is made based on the individual paths replies. If the session is rejected, a message is transmitted over each path that accepted the session to deallocate the resources. If the session is accepted, the optimization phase algorithm (to be described in Section 3.4) that optimizes the resources allocated is initiated.

Discussion We observe here that the amount of resources to be pre-reserved is no greater than the peak rate of one unicast call, usually much smaller than the amount of available resources at a link. This suggests that the transient over-reservation necessary in the allocation phase should have limited impact on resource availability (and is also limited to the time the allocation phase needs to complete). The process of admission control and resource reservation has a transactional form: an atomic action is done beginning with the admission control, based on the peak rate and ending with the actual allocation of resources. This is required in order to prevent inconsistencies between different values of available resources at admission and reservation times. This is also useful in

the case that several (partially) overlapping multicast sessions are established simultaneously. The algorithm is deadlock free because it does not wait for a resource to become available. A rejected connection can be tried later as discussed in Section 5.

3.4 The optimization phase: optimizing the resource allocation in the multicast tree

Once the allocation phase is successfully completed (the multicast session is accepted) the source can start transmitting user data. At the same time, the resource allocation in the multicast tree can be optimized without interfering with the user traffic. In this optimization phase some resources allocated in the allocation phase can be released.

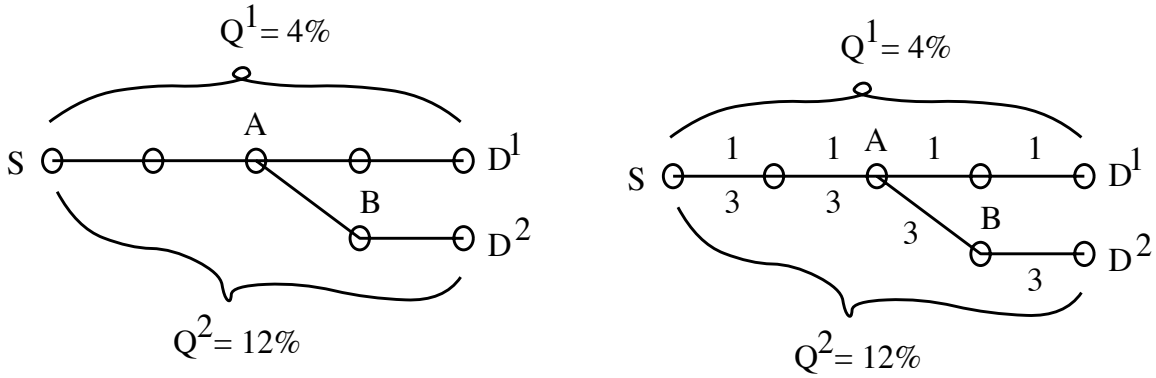
3.4.1 The general algorithm

The need for the resource optimization phase comes from the fact that, after performing the resource allocation, a path P^2 may have more resources reserved than needed because of the requirements of some other path P^1. When this occurs, it should be possible to reduce the resources allocated on the rest of the path P^2 while still meeting its end-to-end QoS requirement.

In Figure 4(a) we have an example of two paths (S, D^1) and (S, D^2) with their end-to-end QoS requirements $Q^1 = 4$, $Q^2 = 12$ sharing a common path (S, A). The even division algorithm yields the local QoS assignments depicted in Figure 4(b). First, for each link on the common path (S, A), the minimum (tightest) QoS (= 1) is assigned. Then, we have that $Q(S, D^2) = 1+1+3+3 = 8 < Q^2$. Thus the QoS allocations at $A-B$ and $B-D^2$ can be relaxed (and the necessary resources reduced). Using again the even division procedure for (A, D), the new local QoS requirements on $A - B$ and $B - D^2$ will be $(12 - 2)/2 = 5$. The final, optimized QoS allocation is shown in Figure 4(c).

More precisely, the algorithm is given in Figure 5. We use the notation $\mathcal{D}_A = \{D \in \mathcal{D} \mid A \in \mathcal{N}(S, D)\}$ for a node $A \in \mathcal{T}$ and $\mathcal{D}_l = \{D \in \mathcal{D} \mid l \in \mathcal{L}(S, D)\}$ for a link $l \in \mathcal{T}$. 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 optimized QoS requirements $(Q_l)_l$ and the corresponding resource allocations $(T_l)_l$ for all links $l \in \mathcal{T}_A$. For each link l outgoing from A in $\mathcal{T}$, GENERAL_OPTIMIZER first minimizes the resources at link l (lines 5-9) and then calls itself recursively for its immediate successor (lines 10-13). The local QoS and resources for l are computed based on the requirements $Q(A, D) = Q(S, D) - Q(S, A)$ for each leaf D downstream l in $\mathcal{T}$. The maximum of these resources is allocated to l, the QoS guarantee is computed for the path (S, B) and GENERAL_OPTIMIZER is applied to the tree rooted in l's destination, B. If the path consists of only one link (lines 2-3), no QoS division is necessary; the resource is simply allocated at that link. $V_{A,D}$ and $V_{S,B}$ are computed in lines 6 and 12 as defined in Section 3.2.2. To optimize the resource allocation for the entire tree $\mathcal{T}$, the procedure call is:

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



a) An example problem

b) The non-optimized solution

c) The optimized solution

Figure 4: An example of resource allocation and optimization

GENERAL_OPTIMIZER(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 $V_{A,D} \leftarrow V_{S,A} \ominus V_{S,D}$
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 $V_{S,B} \leftarrow V_{S,A} \oplus V_{A,B}$
13 GENERAL_OPTIMIZER$(B, \mathcal{D}_B, V_{S,B}, Q(S,B), (V_{S,D})_{D\in\mathcal{D}}, (V_m)_{m\in\mathcal{T}}, (Q(S,D))_{D\in\mathcal{D}}; (Q_l, T_l)_{l\in\mathcal{T}_B})$

Figure 5: The general optimization algorithm

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.

Complexity Analysis We introduce some notation used in subsequent complexity analyses:
$N = |\mathcal{D}|$ is the number of destinations;
$M = |\mathcal{T}|$ is the number of links in the multicast tree;
$L = \sum_{D\in\mathcal{D}} |\mathcal{L}(S,D)| = \sum_{D\in\mathcal{D}} |\mathcal{L}(P^D)|$ is the sum of the number of links contained in each path;
It is easy to see that

$$N \leq M \leq L \leq NM\,. \tag{18}$$

We first address the case of even division without resource limitation and proportional division policies, where COMPUTE_QOS_LINK has a computational complexity of $O(1)$ (see Section 3.2.2). The following is a worst case running time analysis.

Lines 2-4 are executed once for each $D \in \mathcal{D}$, for a total of $O(N)$ times. Lines 9-12 are executed once for each $l \in \mathcal{T}$, for a total of $O(M)$ times. The dominant lines are 6-7 and 8. In the worst case lines 6-7 are executed once for each pair of nodes A, D that has a path (A, D) contained in $\mathcal{T}$, with a computational cost of $O(1)$ each. The aggregate cost will be:

$$\sum_{D\in\mathcal{D}} |\,\mathcal{L}(P^D)\,| = L \tag{19}$$

In line 8 $\min_{D\in\mathcal{D}_l}$ is executed for each outgoing link l from A, with a running time that is $O(|\mathcal{D}_A|)$. Its total cost is:

$$\sum_{A\in\mathcal{T}} |\,\mathcal{D}_A\,| = \sum_{D\in\mathcal{D}} |\,\mathcal{L}(P^D)\,| = L \tag{20}$$

Consequently, GENERAL_OPTIMIZER has a worst case running time $O(N+M+L)$ which is $O(NM)$ from (18).

In the case of even division with resource limitation, a similar analysis applies, with the only difference in the complexity of COMPUTE_QOS_LINK in line 7. We assume that the set $(Q_m^{min})_{m\in(S,D)}$ is sorted (complexity $O(\mid \mathcal{L}(S,D) \mid \log \mid \mathcal{L}(S,D) \mid)$) and then scanned (complexity $O(\mid \mathcal{L}(S,D) \mid)$) in line 3 Figure 1 for each call to COMPUTE_QOS_LINK on the path (A,D) with $A \in \mathcal{N}(S,D)$. We have seen that the loop 3-12 in Figure 1 has a worst case running time of $O(\mid \mathcal{L}(A,D) \mid)$. So, the aggregate complexity of line 7 in Figure 5 is:

$$\sum_{D\in\mathcal{D}} (O(\mid \mathcal{L}(S,D) \mid \log \mid \mathcal{L}(S,D) \mid) + \sum_{A\in\mathcal{N}(S,D)} O(\mid \mathcal{L}(S,D) \mid + \mid \mathcal{L}(A,D) \mid)) = \tag{21}$$

$$\sum_{D\in\mathcal{D}} O(\mid \mathcal{L}(S,D) \mid^2) = O(NM^2)\ . \tag{22}$$

Consequently, GENERAL_OPTIMIZER has a worst case running time $O(N+M+L+NM^2)$ which is $O(NM^2)$ from (18).

3.4.2 An improved algorithm

The repeated application of GENERAL_OPTIMIZER at each link in $\mathcal{T}$ that is shared by several paths may be computationally too expensive. The complexity can be reduced in the case that the QoS division policy exhibits the *uniformity* property that has been described in Section 3.2.3.

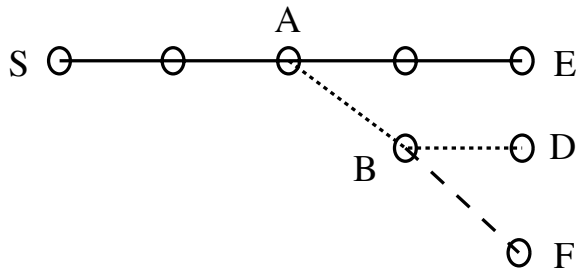


Figure 6: A tree partitioned into critical paths

The idea is that it is not necessary to run the optimization algorithm for each shared link. Let a critical destination for a link be the destination that requires the largest amount of resources on that link. The *uniformity* property ensures that, if D is the critical destination for the link $A \xrightarrow{l} B$, then D is the critical destination for all links on (A,D). If A is the node closest to S with the above property, then we call (A,D) a *critical path.* It is easy to see that critical paths begin only at S or a branching node, and end in a receiver. Furthermore, the tree $\mathcal{T}$ is partitioned into critical paths (see Figure 6 for an example of a partitioned tree).

Given a link l, we can determine the critical destination for l by calling COMPUTE_QOS_LINK on each path sharing l and comparing the path characteristics Q_P as presented at the end of Section 3.2.3. The QoS division computed on a critical path (A, D) for the corresponding QoS requirement $Q(A, D) = Q(S, D) - Q(S, A)$ gives the necessary and sufficient resources for the multicast session $\mathcal{M}$ on that path. As the QoS requirement $Q(S, D)$ is given, we only need $Q(S, A)$ to be able to allocate resources on (A, D). $Q(S, A)$ results from a previous QoS division on another critical path that contains the branching node A. Hence, we can construct an iterative algorithm that processes one critical path at a time, starting with a critical path containing S. For the example in Figure 6 we first optimize resources for (S, E), then compute $Q(S, A)$, continue with optimization on (A, D), compute $Q(S, B)$ and conclude with optimization on (B, F).

IMPROVED_OPTIMIZER(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 $V_{C,D} \leftarrow V_{S,D} \ominus V_{S,C}$
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 $V_{S,B} \leftarrow V_{S,A} \oplus V_{A,B}$
13 $mark(n)$

Figure 7: The improved optimization algorithm

The improved optimization algorithm given in Figure 7 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.

More precisely, the **while** loop in lines 1-13 determines, for each outgoing link l from source S or branching node C, the critical path that contains it (lines 3-6) by comparing in line 6 the path characteristics $(Q_{(C,D)})_{D\in\mathcal{D}_l}$. For each critical path, it computes and allocates the resources (lines 7,9), computes the QoS guaranteed for the upstream parts of the path (line 11), and marks the links of this critical path as processed (line 13). The algorithm terminates when all the links in $\mathcal{T}$ are marked as processed. $V_{C,D}$ and $V_{S,B}$ are computed in lines 4 and 12 as defined in Section 3.2.2.

The algorithm starts with the values for $V_{S,D}, D \in \mathcal{D}$ known from the allocation phase.

Complexity Analysis We first address the case of even division without resource limitation and proportional division policies, where COMPUTE_QOS_LINK has a computational complexity of $O(1)$ and COMPUTE_QOS_PATH applied on path P has a computational complexity of $O(\mid \mathcal{L}(P) \mid)$ (see Section 3.2.2). The following is a worst case running time analysis for the improved optimizer algorithm in Figure 7. Lines 4 and 5, each having a complexity of $O(1)$, are executed for each pair of nodes C, D that have a path (C, D) contained in $\mathcal{T}$ where C is the source S or a branching node. Defining $\mathcal{B}(\mathcal{T})$ as the set of branching node in $\mathcal{T}$, the aggregate running time for lines 4-5 is $O(N^2)$:

$$\sum_{C \in \mathcal{B}(\mathcal{T})} \sum_{D \in \mathcal{D}_C} 1 \leq N^2$$

because

$$\sum_{D \in \mathcal{D}_C} 1 \leq \sum_{D \in \mathcal{D}} 1 = N$$

and because the number of branching nodes in a tree is less than the number of leaves in that tree. Computing the minimum of QoS requirement in line 6 has a complexity of $\mid \mathcal{D}_l \mid$ for each outgoing link l from a branching node, summing to $\mid \mathcal{D}_C \mid$ for each branching node C, giving a total of $O(N^2)$ that follows from the same computation as above. Line 7 is executed once for each critical path (C, E) with a complexity of $O(\mid \mathcal{L}(C, E) \mid)$. Since the set of critical paths forms a partition of $\mathcal{T}$, line 7 has a total complexity of $O(M)$. All the lines in the loop 8-13 are $O(1)$ and are executed for each link in $\mathcal{T}$, yielding a total running time of $O(M)$. Consequently, IMPROVED_OPTIMIZER has a worst case running time $O(N^2 + M)$.

In the case of even division policy with resource limitation, a similar analysis applies, with the only difference in the complexity of COMPUTE_QOS_LINK in line 5 and COMPUTE_QOS_PATH in line 7. We assume that the set $(Q_m^{min})_{m \in \mathcal{T}}$ is sorted (complexity $O(M \log M)$) and then scanned (complexity $O(M)$) in line 3 Figure 1 for each call to COMPUTE_QOS_LINK or COMPUTE_QOS_PATH. We have seen that the loop 3-12 in Figure 1 has a worst case running time of $O(\mid \mathcal{L}(C, D) \mid)$. So, the aggregate complexity of lines 5 and 7 in Figure 7 is:

$$O(M \log M) + \sum_{C \in \mathcal{B}(\mathcal{T})} \sum_{D \in \mathcal{D}_C} (O(M) + O(\mid \mathcal{L}(C, D) \mid)) = O(M \log M + N^2 M) \; .$$

Consequently, IMPROVED_OPTIMIZER has a worst case running time $O(N^2 + M + M \log M + N^2 M)$ which is $O(M \log M + N^2 M)$ from (18).

We conclude that, in the case of even division without resource limitation and proportional division policies, IMPROVED_OPTIMIZER reduces the worst case running time to $O(N^2 + M)$ from $O(NM)$ for GENERAL_OPTIMIZER. In the case of even division with resource limitation, IMPROVED_OPTIMIZER reduces the worst case running time to $O(N^2 M + M \log M)$ from $O(NM^2)$ for GENERAL_OPTIMIZER. These reductions are important since, in general, the number of leaves (N) of a tree is significantly less than the number of links (M) in that tree.

3.5 Joining and leaving a session

In this section we consider the dynamic multicast problem where receivers can join or leave the session anytime during the life of the session. As mentioned in Section 3.2.3, since join events can be non simultaneous, only even division without resource limitations is *uniform*. We assume that the multicast session $\mathcal{M}$ is already established with the necessary resources optimized. All the computations for resource allocation and optimization are done in a single place, which holds also all the necessary network data. This place is usually the sender.

Join We assume that all the information obtained during the execution of the allocation and optimization phases is still available. Specifically we require the set $(Q(S,A))_{A\in\mathcal{T}}$ and $(T_l)_{l\in\mathcal{T}}$ where T_l is the resource allocated on l for this multicast session. Also we use the convention that $T_l = 0$ for $l \notin \mathcal{T}$. R_l represents the amount of available resources on l. We describe modified versions of the allocation phase presented in Section 3.3 and the optimization phase from Section 3.4.2.

The allocation phase. We assume that the path $P^E = (S, E)$ for the joining receiver E is given to us by a routing algorithm. For this path, check if the end-to-end QoS $Q(S, E)$ can be met by allocating either $(T_l + R_l)$ or the maximum resources (peak rate) required by the session at each link on the path, whichever is smaller. If the check is negative, reject the join. Else accept it and, for the path P^E, divide $Q(S, E)$ among the links on the path and compute the resources needed $(T_l^E)_{l\in\mathcal{L}(P^E)}$. Assign for $l \in \mathcal{L}(P^E)$, $T_l' = \max(T_l, T_l^E)$ and allocate resources $R_l \leftarrow R_l - (T_l' - T_l)$.

The optimization phase. If the QoS division is not *uniform* (e.g. even division with resource limitations or proportional division), the GENERAL_OPTIMIZER algorithm described in Section 3.4 has to be called starting in S and only for the subtree of $\mathcal{T}$ that has common links with P^E. As new values Q_l^{new} are computed for the links $l \in \mathcal{L}(P^E)$, a new allocation is done only if $Q_l^{new} < Q_l^{old}$ and so $Q(S,A)^{new} \leq Q(S,A)^{old}$ is true for all $A \in \mathcal{N}(P^E)$, where the 'old' QoS values are guaranteed for the existing multicast session. GENERAL_OPTIMIZER has to be called recursively only for those subtrees rooted in A that have $Q(S,A)^{new} < Q(S,A)^{old}$ because there is no need to optimize the subtrees that have $Q(S, A)$ unchanged.

If the QoS division is *uniform* (e.g. even division without resource limitations), the algorithm is similar to IMPROVED_OPTIMIZER described in Section 3.4.2. Using the same argument as above, we start by finding the node A closest to S where the QoS requirement for P^E are more stringent than those already guaranteed for the existing multicast session, $Q(S,A)^{new} < Q(S,A)^{old}$. Then we unmark the links of the subtrees rooted in A and execute the steps 2-13 of IMPROVED_OPTIMIZER.

Leave Destination E leaves the multicast session. All the resources for the part of P^E that is not shared with other paths in $\mathcal{T}$ are deallocated. For the rest of P^E the resources on the links do not have to be modified because they belong to already optimized paths.

4 A complete example

In the following we will construct an example of a network specifying its characteristics (the link scheduling policy is Generalized Processor Sharing, the sessions' arrival process is packetized voice modeled by on/off Markov fluids, the QoS metric is loss probability) and show how various mechanisms proposed in this paper can be implemented. Some of the computations that follow are based on results presented in [KWC93, GAN91].

4.1 The network and the link scheduling policy

The network consists of nodes that use Generalized Processor Sharing (GPS) as the link scheduling policy among L classes of sessions. A class is characterized by a unique QoS that is guaranteed for all sessions that are admitted in that class (in our case loss probability, see Subsection 4.2). The FIFO policy is used among sessions within a class. Given link l, the coefficients used as GPS class weights are $(\phi_{l,k})_{k=1,..,L}$ such that $\sum_{k=1}^{L} \phi_{l,k} = 1$. All links have identical characteristics (service rate, QoS of each class, etc.) but not necessarily the same GPS coefficients ("class allocation", ϕ_k). These GPS coefficients will be dynamically adjusted to accommodate various class loads, as described in Section 4.4.

4.2 The local QoS, composition and end-to-end division

All the sessions within class k will share a buffer of capacity B_k. Due to the use of FIFO for scheduling within a class, one can show (see [KWC93, GAN91]) that the loss probability is the same for all the sessions in this class, and has the following expression:

$$\lim_{B_k \to \infty} \frac{1}{B_k} \log \Pr\left(X_k > B_k\right) \leq \; - \delta_k \tag{23}$$

that gives an asymptotic exponential upper bound (valid for large values of B_k):

$$\Pr(X_k > B_k) \leq e^{-\delta_k B_k} \quad . \tag{24}$$

where X_k is the backlog (queue length) and δ_k is the exponential decay rate that links the loss probability to the effective bandwidth condition in (26). From now on we will assume that the set $(\delta_k)_{k=1,..,L}$ is fixed and identical for all links. Also we will assume $\delta_k < \delta_{k+1}, \quad k = 1, .., L-1$, i.e. class 1 has the highest loss probability. We will also assume that the end-to-end loss probabilities are $\ll 1$. In this case, given a path $P = (l_1, .., l_n)$, the composition of local loss that gives a good bound for end-to-end loss is the sum, as seen in Section 3.2.1:

$$\sum_{i=1}^{n} \Pr(X_{i,k_i} > B_{k_i}) < \sum_{i=1}^{n} e^{-\delta_{k_i} B_{k_i}} < Q(A_0, A_n) \quad . \tag{25}$$

The reverse operation, the division of QoS, can thus be stated as: given an end-to-end QoS $Q(A_0, A_n)$ for the path P, choose a set $(k_i)_{i=1,..,n}$ (the class to admit the session into at each link) such that equation (25) is satisfied. The QoS division will be done as described in Section 3.2.2.

4.3 The sessions' arrival processes

We will take as the source traffic for a session packetized voice that can be modeled as i.i.d. on/off Markov fluids (see [AMS82] for an early description of this model). Such a traffic model captures readily the behavior of bursty traffic and has been used in the literature ([Bra68, HL86, YKTH93]) to model packetized voice; we use it here also due to its computational simplicity in constructing our example. In this model a source can be in two states: during the "on" period the source is sending data with a constant rate of r and during the "off" (silence) period the source is not sending anything. The durations of the "on" and "off" periods are exponential random variables with means $1/\mu$ and $1/\lambda$ respectively. The session admission and resource allocation presented in Section 4.4 uses the theory of effective bandwidth from [KWC93, GAN91, dVCW93]. The expression for the effective bandwidth of an on/off Markov fluid is:

$$\alpha(\delta) = \frac{\delta r - \mu - \lambda + \sqrt{(\delta r - \mu + \lambda)^2 + 4\lambda\mu}}{2\delta} \quad . \tag{26}$$

where δ is a parameter that characterizes the QoS (loss probability) to be guaranteed, as described in Section 4.2.

Also we know (e.g. from [dVCW93]) that, because we will take $r < c$ (c is the maximum processing rate (capacity) of the server (link)), the effective bandwidth of the departure process is equal to that of the arrival process (on/off Markov fluid). Consequently the session is characterized by the effective bandwidth given in (26) at all the links of the multicast tree. In conclusion, combining formulas 24 and 26, and denoting $Q_k = \Pr(X_k > B_k)$ as the local QoS (loss probability) we derive a closed-form expression for the function F introduced in Section 3.1 :

$$F(Q_k) = \frac{\frac{-\log Q_k}{B_k} r - \mu - \lambda + \sqrt{(\frac{-\log Q_k}{B_k} r - \mu + \lambda)^2 + 4\lambda\mu}}{2\frac{-\log Q_k}{B_k}} \quad . \tag{27}$$

4.4 Session admission and resource allocation

To simplify admission test computations we will use a conservative approximation of resource utilization in a server using the GPS queue discipline. We will assume that the GPS server with bandwidth c is decoupled into L sub-servers corresponding to the L classes, each having a bandwidth of $(\phi_k c)_{k=1,..,L}$. This assumption is conservative in the sense that the bandwidth of a class that is idle at one moment cannot be used by the other classes. The session admission criterion for each sub-server k is:

$$\Pr(X_k > B_k) \leq e^{-\delta_k B_k} \quad \text{iff} \tag{28}$$

$$N_k \alpha(\delta_k) < \phi_k c \quad , \tag{29}$$

where N_k is the number of sessions in class k. We take as the session admission criterion for the GPS server

$$\sum_{k=1}^{L} N_k \alpha(\delta_k) < c \quad . \tag{30}$$

It is clear that, if (30) is satisfied, then

$$\exists(\phi_k)_{k=1,..,L}\ ,\ 0 \leq \phi_k \leq 1\ ,\ \sum_{k=1}^{L} \phi_k = 1$$

such that (29) is satisfied for all $k = 1, .., L$ and so the QoS is guaranteed for all L classes as stated in (28).

4.5 A numerical example

We present a short numerical example of call admission, resource allocation and optimization for a simple network and three classes of loss.

The session's arrival process is a Markov fluid that models packetized voice encoded using ADPCM with the following parameters often used in the literature ([Bra68, SW86, HL86]):

- mean ON time $1/\mu = 0.352s$
- mean OFF time $1/\lambda = 0.650s$
- peak rate $r = 32Kb/s$

Each class has a buffer capacity of $B = 30Kb$ for each class and the loss probabilities for the classes are $P_1 = 0.017$, $P_2 = 0.005$, $P_3 = 0.001$, The effective bandwidths associated with each of these QoS classes for the voice mode described above is listed in the following table, computed using the effective bandwidth formula from (26), and the loss probability bound from (28):

$\alpha(\delta_j)$ $[Kb/s]$	P_j
18.818	0.017
20.602	0.005
22.431	0.001

Let us now consider the network in Figure 8(a) where we would like to establish a multicast voice session from S to D^1 and D^2, with the loss requirements $Q(S, D^1) = 0.002$ and $Q(S, D^2) = 0.001$. Clearly the multicast session can be admitted using for example the lowest loss probability class in all links $(4 * 0.1\% < Q(S, D^2) < Q(S, D^1))$. Now applying the even division policy for QoS classes as in Section 3.2.2 we get the loss class allocation as in Figure 8(b). Finally, using the optimization algorithm, we find that we can relax the loss class in D^1 from 0.005 to 0.017 as shown in Figure 8(c), thus allowing a reduction in resource allocation from $20.6Kb/s$ to $18.81Kb/s$. The final resource allocation is shown in Figure 8(d).

4.6 Simulations

The example above is applied 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 (Figure 9). We assume that only 5% of the T3 capacity is used for voice multicast applications, so

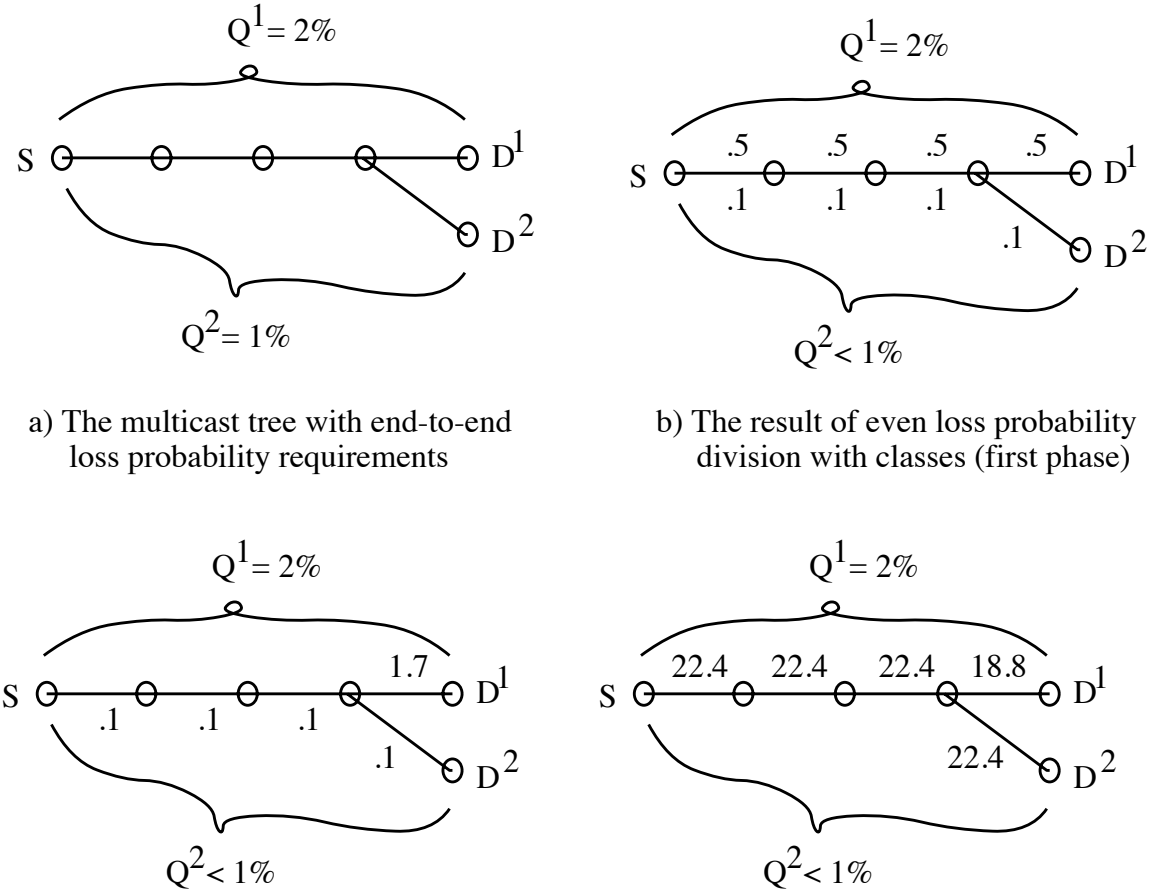


Figure 8: The numerical example

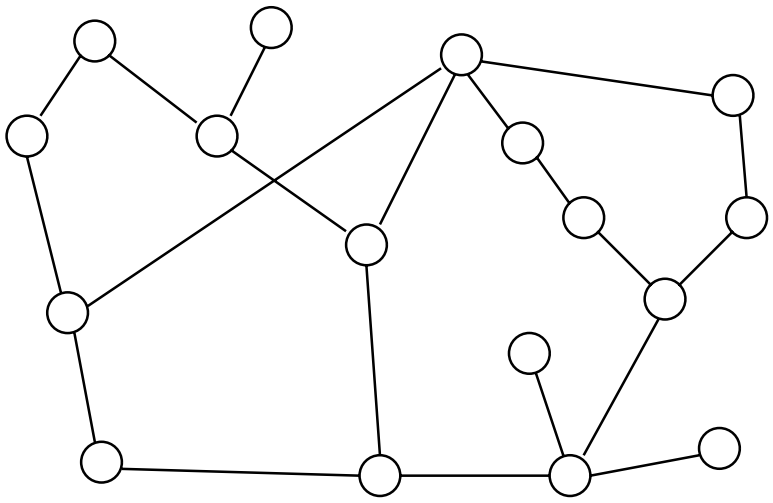


Figure 9: The T3 MBone

each unidirectional link will have a bandwidth of $c = 2.25Mb/s$ allocated for multicast calls. Multicast calls are generated according to a Poisson process with parameter λ and their durations are exponentially distributed with mean $1/\mu$. $\rho = \lambda/\mu$ characterizes the load offered to the network, i.e. the average number of multicast calls that would exist at any time in an infinite resource network. Each multicast call has its source and destinations chosen with equal probability from the nodes of the network. The number of destinations is also uniformly distributed in the range $1, 2, .., 16$. The end-to-end loss probability for each source-destination pair is also uniformly distributed in the range $[10^{-6}, 10^{-1}]$. This somewhat wide range has been chosen to compensate for the fact that we didn't simulate the whole MBone which is a network much larger than the T3 backbone used here. 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 over-reservation of bandwidth due to admission in the next QoS class is less than 6%, which is also confirmed in our simulations.

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 optimization phase (Section 3.4). Figure 10 shows the call rejection probability as a function of offered load ρ, and Figure 11 shows the relative difference in the performance of optimized Even, non-optimized Proportional and optimized Proportional with respect to non-optimized Even as a function of offered load. We can see that both the optimized Even division and non-optimized Proportional division bring significant gain (lower blocking probability) compared to the non-optimized Even division. This gain is further increased if we combine the two in the optimized Proportional division, as seen in Figure 11 that plots the relative gain.

5 Discussions, conclusions and future work

We have developed solutions to the problems of call admission control and resource reservation 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 optimization phase is responsible for refining it 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 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 optimized Proportional QoS division policy provides a 30% to 70% decrease in call rejection probability compared to other policies.

We have made several assumptions in our work that can be easily relaxed. For example, although we focussed on a QoS metric based on loss probability, all of the algorithms apply equally as well in the case that the QoS metric is maximum delay. Furthermore, the algorithms can be applied

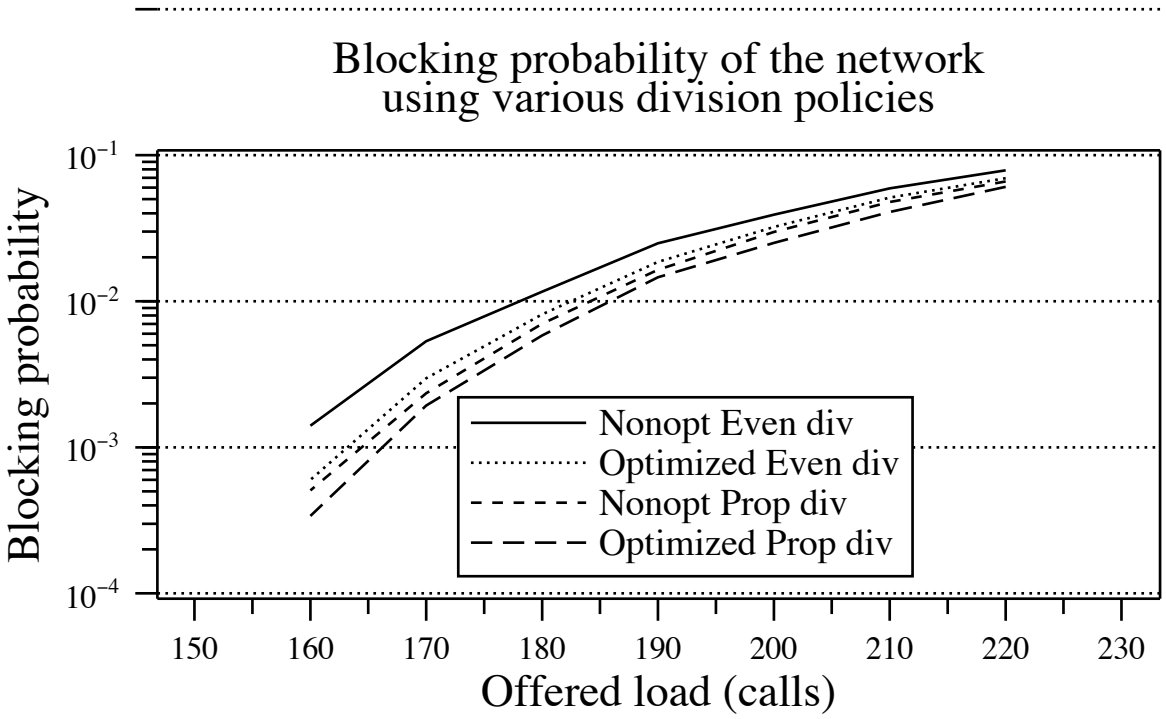


Figure 10: The blocking probability

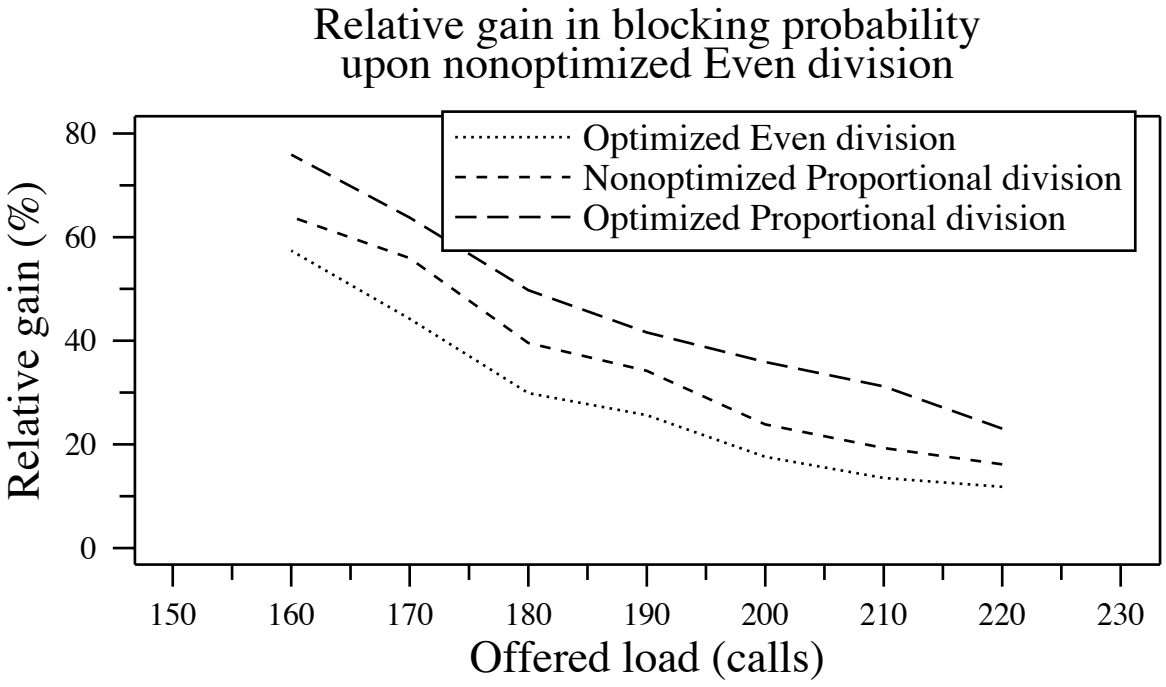


Figure 11: The relative gain

to QoS metrics such as the probability that the end-to-end delay exceed a known quantity and a bound on delay jitter with minor modifications.

We also assumed that a multicast session can be accepted only if all destinations can be guaranteed their QoS requirements. It should be clear that the algorithms can be modified to support other QoS semantics such as,

- as many destinations as possible are accepted with the option that the rest join as the network resources permit;
- acceptance occurs only if a quorum destinations can have their QoS requirements satisfied.

Our algorithms for resource optimization 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 resource optimization are distributed in the nodes of the multicast tree). This makes the object 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. Another 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

[AMS82] D. Anick, D. Mitra, and M. M. Sondhi. Stochastic theory of a data-handling system with multiple sources. *Bell System Technical Journal*, 61, 1982.

[And93] David A. Anderson. Metascheduling for Continuous Media. *ACM Transactions on Computer Systems*, 11(3), August 1993.

[BFC93] Tony Ballardie, Paul Francis, and Jon Crowcroft. Core Based Trees. In *ACM SIGCOMM '93*, pages 85–95, September 1993.

[Bra68] P. T. Brady. A statistical analysis of on-off patterns in 16 conversations. *Bell System Technical Journal*, 47:73–91, January 1968.

[Cru91] Rene L. Cruz. A Calculus for Network Delay, Part I: Network Elements in Isolation. *IEEE Transactions on Information Theory*, 37(1), January 1991.

[DC90] Stephen E. Deering and David R. Cheriton. Multicast Routing in Datagram Internetworks and Extended LANs. *ACM Transactions on Computer Systems*, 8(2):85–110, May 1990.

[DL93] Matthew Doar and Ian Leslie. How Bad is Naïve Multicast Routing ? In *IEEE INFOCOM '93*, pages 82–89, 1993.

[dVCW93] G. de Veciana, C. Courcoubetis, and J. Walrand. Decoupling Bandwidths for Networks. Technical Report M93/50, UCB/ERL, June 1993.

[EMM93] Wolfgang Effelsberg and Eberhard Müller-Menrad. Dynamic Join and Leave for Real-Time Multicast. Technical Report TR-93-056, Tenet Group, U.C. Berkeley and ICSI, October 1993.

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

[FV90] Domenico Ferrari and Dinesh Verma. A Scheme for Real-Time Channel Establishment in Wide-Area Networks. *IEEE Journal on Selected Areas in Communications*, 8(3), April 1990.

[GAN91] 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.

[GG92] Roch Guérin and Levent Gün. A Unified Approach to Bandwidth Allocation and Access Control in Fast Packet-Switched Networks. In *IEEE INFOCOM '92*, 1992.

[GGP94] L. Georgiadis, R. Guérin, and A. Parekh. Optimal multiplexing on a single link: Delay and buffer requirements. Technical Report Research Report RC 19711, IBM, T.J.Watson Research Center, August 1994.

[GH91] R.J. Gibbens and P.J. Hunt. Effective bandwidths for the multi-type UAS channel. *Queuing Systems*, 9, 1991.

[HL86] H. Heffes and D. M. Lucantoni. A Markov modulated characterization of packetized voice and data traffic and related statistical multiplexer performance. *IEEE Journal on Selected Areas in Communications*, SAC-4:856–868, September 1986.

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

[Kel91] F. P. Kelly. Effective bandwidths at multi-class queues. *Queueing Systems*, 9, 1991.

[KWC93] G. Kesidis, J. Walrand, and C. S. Chang. Effective Bandwidths for Multiclass Markov Fluids and Other ATM Sources. *IEEE/ACM Transactions on Networking*, August 1993.

[MESZ94] Danny A. Mitzel, Deborah Estrin, Scott Shenker, and Lixia Zhang. An Architectural Comparison of ST-II and RSVP. In *IEEE INFOCOM '94*, 1994.

[NT94] Ciro A. Noronha Jr. and Fouad 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.

[Par92] 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.

[Sch92] Henning 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.

[SW86] K. Sriram and W. Whitt. Characterizing superposition arrival processes in packet multiplexers for voice and data. *IEEE Journal on Selected Areas in Communications*, SAC-4:833–846, September 1986.

[Top90] C. Topolcic. *Experimental Internet Stream Protocol: Version 2 (ST-II)*. Internet RFC 1190, October 1990.

[Wax88] Bernard M. Waxman. Routing of Multipoint Connections. *IEEE Journal on Selected Areas in Communications*, 6(9):1617–1622, December 1988.

[Wax93] Bernard M. Waxman. Performance Evaluation of Multipoint Routing Algorithms. In *IEEE INFOCOM '93*, pages 980–986, 1993.

[YKTH93] David Yates, James Kurose, Don Towsley, and Michael G. Hluchyj. On per-session end-to-end delay distributions and the call admission problem for real-time applications with QOS requirements. In *ACM SIGCOMM Symposium on Communications Architectures and Protocols*, September 1993.

[YS93] Opher Yaron and Moshe Sidi. Performance and Stability of Communication Networks via Robust Exponential Bounds. *IEEE/ACM Transactions on Networking*, 1(3), June 1993.

[ZDE+93] Lixia Zhang, Stephen E. Deering, Deborah Estrin, Scott Shenker, and Daniel Zappala. RSVP: A New Resource ReSerVation Protocol. *IEEE Network*, 7(5):8–18, September 1993.

[ZTK94] Zhi-Li Zhang, Don Towsley, and Jim Kurose. Statistical Analysis of Generalized Processor Sharing Scheduling Discipline. In *ACM SIGCOMM '94*, 1994.