Scalable resource reservation for multi-party real-time communication

Amit Gupta, Wingwai Howe,
Mark Moran, Quyen Nguyen
{amit,whowe,moran,nguyenq}@tenet.Berkeley.EDU
Tenet Group
University of California at Berkeley, &
International Computer Science Institute

TR-94-050
October 1994

Abstract

Current approaches to supporting real-time communication allocate network resources either to individual connections, or to aggregates of connections, based on type of traffic, protocol, or performance requirements. The first approach provides well-defined performance guarantees that are independent of other network traffic. The second approach may achieve higher utilization of network resources, but the expected performance is less well-defined since it is dependent on the behavior of unrelated (possibly unknown) connections. *Resource sharing* is a new approach that exploits known relationships between *related* connections to allow network resources to be shared without sacrificing well-defined guarantees. Most importantly, for large conferences with a bounded number of *concurrent* speakers, resource requirements do not increase with the number of potential speakers. Therefore, resource sharing is an important tool for providing real-time performance guarantees for large conferences. This paper presents a fully distributed technique for using resource sharing to provide real-time guarantees in a general internetworking environment. The technique is described in the context of its implementation in the next generation of the Tenet real-time protocols. However, the underlying principles are equally applicable to other communication paradigms and techniques. A companion report presents the results of simulation experiments; the simulations show that resource sharing leads to large gains in connection acceptance rates and a significant reduction in computational overhead associated with admission control for real-time communication.

1 Introduction

Many classes of applications, including distributed multimedia group communication [13] and traditional distributed processing, require or benefit from a network communication service that provides well-defined performance guarantees. A number of protocols and schemes have been proposed to provide real-time communication services [2, 6]. These schemes are usually connection-oriented, in that they allocate network resources (e.g., bandwidth, buffers, etc.) along the path data packets will travel.

Traditional real-time network systems (e.g., [4]) may over-allocate resources for two reasons: (1) they allocate resources based on a worst-case prediction of the actual traffic; and (2) they treat traffic on different connections independently when determining their resource requirements. One technique to improve utilization (and hence the connection acceptance rate) is to measure the actual traffic characterization of individual connections and to modify the connection traffic characterization dynamically [16]. Another technique uses performance measurements over aggregations of connections to predict future performance [2]. The first approach still over-estimates aggregate resource requirements, since it fails to capture the important relationships between connections (e.g. in a conference, usually only one speaker is active at a time). The second approach will indirectly capture these relationships, but it fails to provide protection against unrelated traffic, and depends on the assumption that current behavior adequately predicts future behavior for all connections in the aggregate. This assumption may not be valid when a single connection can have a significant effect on the performance of other connections, e.g., over low-capacity links. In addition, when measurements are not available (e.g. when a connection is first established), this approach must fall back to use independent traffic characterizations, as in the first approach.

In this paper, we propose *resource sharing* as a "middle ground", by which *related* connections can *share* resource allocations in a controlled manner, so that network utilization is improved and performance guarantees of established connections are achieved. Resource sharing differs from techniques that rely on *statistical multiplexing* of (unrelated) network traffic, in that the network client specifies how traffic from related connections is multiplexed. As long as the aggregate traffic of these related connections does not exceed this specification, the network service guarantees that well-defined performance bounds will be met for individual channels. As the client specifies the related connections and their aggregate traffic, all sharing is completely client-controlled. Most importantly, performance guarantees are *not* dependent on the behavior of unrelated network traffic. Resource sharing is an important technique to provide well-defined performance guarantees for most large-scale, multi-party communication paradigms.

We have organized this paper in the following manner. In Section 2, we provide the background for the current work with a brief description of the Tenet protocol framework. Section 3 presents a simple example to motivate resource sharing. In

Section 4, we discuss the mechanisms that we have designed for efficiently supporting resource sharing in the next generation of the Tenet real-time protocol suite (known as Suite 2), which Section 5 illustrates with a simple example. We discuss the interactions of resource sharing with other service/protocol components in Section 6. Section 7 describes the related work by other researchers, and we conclude this paper with suggestions for future work in Section 8.

2 Background

In this section, we provide a very brief introduction to the Tenet protocols; interested readers can find a more detailed explanation in [4, 11]. It may be noted that while the current discussion is in the context of the Tenet protocols, the underlying ideas are equally applicable to other communication schemes and paradigms.

In the second generation of the Tenet protocols [7], the basic abstraction is the multicast real-time channel—a simplex, one-to-many connection at the network-layer. A conference with M sources and N destinations consists of M multicast channels to the same set of N destinations, where the destination set is referred to as the *Target Set* for that conference. A channel is characterized by traffic and performance specifications given by the network service client(s). The traffic specification bounds the data rate at which the channel may inject traffic into the network. The performance specification describes the performance bounds that the network guarantees to meet for compliant clients. A typical set of performance parameters includes bounds on the end-to-end packet delay (delay bound), on the variation in the end-to-end packet delay (delay jitter bound), on the probability that the delay bound will be met (timeliness bound) and on the packet loss rate due to buffer overflow (loss bound). The Tenet approach is connection-oriented and reservation-based: before data can be transmitted on a real-time channel, the channel must be established (i.e., resources must be allocated for the channel at each server along its route), so that the guarantees are supported. The required resources may not be available along one or more branches of the multicast tree, resulting in a partial failure of channel establishment.

We will now describe source-initiated channel establishment; channel establishment can also be receiver-initiated. Channel establishment is a fully-distributed two-pass process. In the first (forward) pass, a message from the source visits each node on the multicast route of the channel. At each intermediate node, the network makes tentative reservations and forwards the establishment message to the next node. In the second (reverse) pass, reply messages flow from all destinations to the source along the same route as was used for the forward pass, but in the reverse direction. The resource reservations are committed on the reverse pass.

3 Motivation for resource sharing

In this section, we motivate resource sharing with a simple example. We present a simplified tele-conferencing scenario to motivate the need for resource sharing. Consider the simple conference scenario presented in Figure 1, where a conference is set up among A, B and C (X is an intermediate node or router). Only the two multicast channels from A and from B are shown in Figure 1.

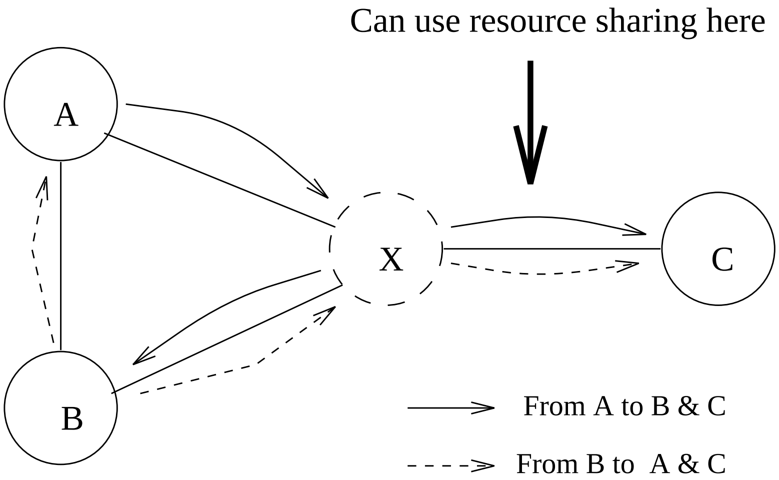

Figure 1: The 3-user Conference Example

Due to the cooperative nature of the conference, it is reasonable to require that only one person speaks at any time. Indeed, in an orderly meeting only one person speaks at any time; two persons speak simultaneously only when they try to *get the floor*; clearly, this situation lasts for but a short period of time, and it should be acceptable if the performance degrades during that time period.

Consider the link X-C; the two multicast channels (from A and from B) can share the resources on the link X-C. Without resource sharing, the network will make independent reservations for the two channels, and wastefully over-allocate resources on the link X-C by 100%. Under resource sharing, the client will inform the network that the two channels are both part of the same conference, and that the aggregate traffic on the channels will not exceed the traffic due to one source. The network can use this information to limit the resource reservations on the link X-C.

This example illustrates a scenario where only one source is active at any time in the conference. In general, we can have up to n concurrently active sources. We define the *maximum concurrency* for a conference as the maximum number of concurrently active sources; in the example above, the maximum concurrency is equal to one.

While the above example illustrates the need for resource sharing in a simple conference scenario, it should be noted that resource sharing is equally useful in other real-time multi-party scenarios such as panel discussions, distributed seminars and so on. For these multi-party applications, the maximum concurrency is usually smaller than the number of senders and, most significantly, *does not increase with the number of participants.* In such cases, resource sharing leads to more efficient use of network resources. In fact, since in most cases we expect the maximum concurrency to remain fairly small even when the number of participants increases dramatically, resource sharing enables better scalability for large conferences.

4 Mechanisms for resource sharing

The key motivation for resource sharing is to exploit the known behavior of related channels in order to reduce the aggregate network resources allocated to these channels. To be attractive, resource sharing must give network clients the same performance guarantees that they would have received without resource sharing. The mechanisms we have devised are completely distributed; hence, they do not restrict the scalability of communication, and are robust in the presence of node and link failures. Indeed, simulation results indicate that resource sharing improves the scalability of communication [8]. Three types of mechanisms are required:

- *Client-service interface:* The network client must inform the network of sharing relationships between channels. This interface defines the contractual agreement between the client and the network.

- *Admission control tests:* The network admission control tests may use the information supplied by the client to perform local admission control tests on a group of channels, rather than on each channel individually.

- *Protection:* The network must ensure that network resources consumed by the channels in a group do not exceed the resource allocation of the group.

4.1 Client-service interface

To allow the network to share resource allocations between related channels, the client must specify three kinds of information:

- *A list of related channels that may share resources.* Two of the authors have previously defined the *channel groups* abstraction to enable clients to specify inter-channel relationships to the network [9]. To specify a list of channels that may share resources, we define a channel group with a *resource sharing* relationship. Individual channels then join the group to share resources with other member channels.

- *Resource requirements for each group.* Our assumption is that at any given time the actual resources required by all group members will not exceed the resources allocated if the channels were treated independently. To benefit from this situation, the client must specify the maximum aggregate resource requirements for the group of channels. Two approaches can be used:

1. The client can specify directly the maximum aggregate resource requirements of the group of channels, or
2. The client can specify the maximum concurrency between channels, and the network can compute a maximum resource requirement for the aggregate along each link.

We have chosen the first approach, because, in the case where the maximum concurrency between channels is greater than one (e.g. when several video channels are displayed during a seminar), the client may specify a resource requirement for the combined streams that takes into account gains from statistical multiplexing between *related* channels. In the second alternative, the network does not know how traffic on separate channels may combine, and thus must treat channels independently of one another.

- *When the group requirements should be used.* In the case where the maximum concurrency is greater than one, the group requirements may be significantly greater than the resources required by any individual channel. Therefore, the client must indicate to the network when to use the group specification rather than the individual ones. We take the simplest approach and specify a *sharing threshold* that corresponds to the maximum concurrency of the group. When the number of member channels on a link equals or exceeds the threshold, the group specification (described above) is used. Before that time, resources are reserved for each channel independently of the others in the group.

4.2 Admission control

The admission control tests determine if a new channel can be admitted without potentially violating the guarantees given to established channels. As described in Section 2, the Tenet protocols utilize a fully-distributed technique for connection establishment and admission control. The modifications to support resource sharing maintains this fully-distributed property. The key change is that the group resource allocation is used in admission control tests instead of the individual (per-channel) allocations when the number of member channels at a server equals or exceeds the threshold. After the sharing threshold has been reached, *no admission tests need be performed to admit additional member channels.*

In the Tenet scheme, three resources are reserved: link throughput, buffer space, and scheduling priority (which determines queueing delay). To simplify changing from individual specifications to the group specification, all channels are given the same local delay bound during establishment, even when the scheduling discipline would allow us to give separate delay bounds per channel (such as with Earliest Due Date scheduling [10]). If a given channel requires a tighter delay bound than the bound given to group members, that channel can be established separately from the group at one or more servers.

Table 1: Link throughput allocation for a sharing group vs. number of channels

No. of channels	Threshold	Sum of channel specs. (Mbps)	Group spec. (Mbps)	Reservation (Mbps)
1	3	1.5	4.0	1.5
2	3	3.0	4.0	3.0
3	**3**	**4.5**	**4.0**	**4.0**
4	**3**	**6.0**	**4.0**	**4.0**
...	...	...	...	...
n	**3**	**n * 1.5**	**4.0**	**4.0**

Table 1 shows an example of the throughput allocated to establish channels from a sharing group at an arbitrary server. In this example, we assume that each channel requires 1.5 Mbps throughout, the sharing threshold is 3, and the maximum throughput required by any three channels is 4.0 Mbps (because of the manner in which the streams are known to multiplex). The table shows the total throughput allocation for the channels as the number of member channels established through the server increases. When the first channel is established, the threshold has not been reached, so 1.5 Mbps is reserved according to the individual specification of the channel. Likewise, the second channel reserves 1.5 Mbps according to its individual specification. The third channel, however, reaches the threshold, so the allocation is changed to the group specification. As can be seen in Table 1, once the threshold has been reached, no tests have to be performed for new channels in the group. This simple example shows how resource sharing improves scalability for large conferences.

For example, if the Rate-Controlled Static Priority (RCSP) scheduling algorithm [19] is used, a channel is assigned a static priority and admission control algorithms ensure that a pre-specified delay bound is met for each priority level. To implement resource sharing, we substitute the group specification for the individual channel specifications in some or all of the tests. In our implementation of resource sharing for servers using RCSP scheduling, link throughput is allocated to the entire channel group according to the group specification. Therefore, all channels of the group receive the same bound on queueing delay.

4.3 Protection

In order to provide guarantees, we must ensure that each channel can use the resources that have been allocated for it. The rate control and scheduling routines do the *policing* that provides this protection. The Tenet protocols protect the resource allocations of real-time channels (i.e. channels that make resource reservations) by giving them higher priority than non real-time channels, and by ensuring that no real-time channel exceeds its resource allocation. The main mechanism for policing real-time channels is rate control, i.e. the network ensures that the traffic for a channel does not exceed its specification. Scheduling priority is protected automatically by the scheduling algorithm, and buffer space allocations are protected by allocating buffers to real-time channels.

To provide protection in the presence of resource sharing, we must provide the same level of policing on group aggregate traffic. To meet this requirement, we allocate resources to the group: when the group specification is in effect, all channels in a sharing group share common resources. Rate control and scheduling are performed by treating all traffic from member channels as belonging to a "super channel" that must obey the group specification. Only one addition is required to support resource sharing: when the group threshold has been reached in a server, rate control and scheduling are performed using the group allocation rather than the allocation for the individual channel. To implement this change, we introduced an indirection from the channel table to the resource allocation used by the rate control and scheduling algorithms. The algorithms themselves do not change. The behavior is shown in Figure 2.

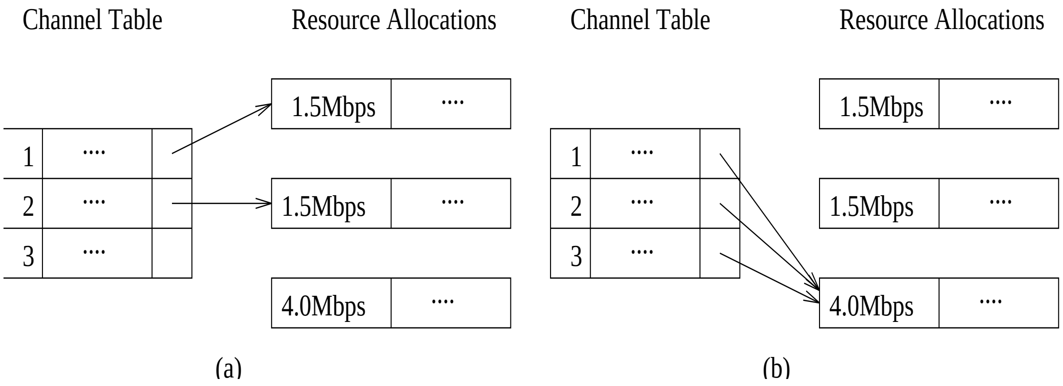

Figure 2: Implementation of rate control to support resource sharing using (a) individual allocations; (b) group allocation.

5 Example

In this section, we illustrate, with the help of a simple example, the various aspects of resource sharing described in the previous section. For simplicity, we consider a simple conference in a small tree-topology network shown in Figure 3.

We assume each conference participant ($P1, P2, ...P6$) is the source for one video channel of 1.5Mbps to all other participants; the maximum aggregate resource requirement for all channels is 4.0 Mbps; and the sharing threshold is 3. In this case, the required resource allocations are as given in Table 1. For simplicity, we also assume that all requests are made by a single conference organizer, who could exist anywhere in the network.

To set up resource sharing, the conference organizer performs the following actions:

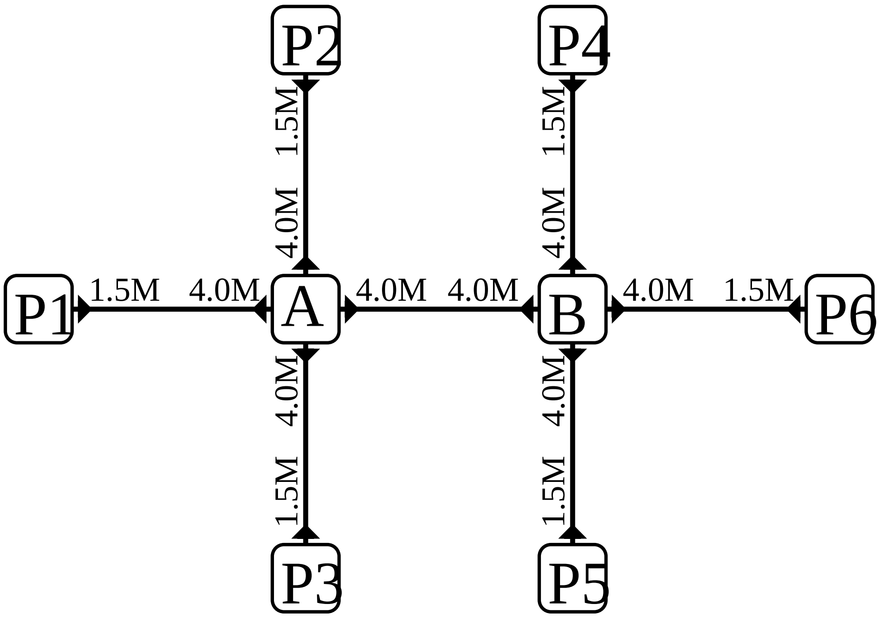

Figure 3: Resource allocations for the 6-user conference

- Request the network to create a Target Set ($TS1$), and to add each participant to the Target Set.
- Request the network to create one channel from each participant to $TS1$; all channels are 1.5 Mbps.
- Request the network to create a sharing group (say $SG1$) with aggregate resource allocation 4.0 Mbps and threshold of 3, and add the channels to the group.
- Request the network to establish these channels.

Below we describe the admission control process for link $B \rightarrow P6$. During establishment, similar computations take place at other links. At this server, five admission control requests arrived for this conference—one for each channel from participants $\{P1, ...P5\}$ to $P6$. The admission control actions for each of the establishment requests at this server is listed below:

- **First request arrives**: The admission control system notes that the request is for the group $SG1$, and that the threshold has not been reached. Therefore, 1.5 Mbps is reserved for the channel according to the individual channel specification.

- **Second request arrives**: The admission control system again notes that the request was for the group $SG1$, and that the threshold has not been reached. Therefore, another 1.5 Mbps is reserved for the channel according to the individual channel specification.

- **Third request arrives**: Since the threshold has been reached, the allocation is changed to the group specification of 4.0 Mbps. The protection and traffic policing is as shown in Figure 2.

- **Fourth and fifth requests**: Since the threshold has been reached, no tests have to be performed and the requests are immediately accepted.

6 Interactions with other components

In this section, we will discuss the interactions of resource sharing mechanisms with three other components of the real-time communication scheme: the local admission control mechanisms, the routing system, and the mechanisms for supporting advance reservation of network resources.

6.1 Interactions with local admission control

With resource sharing, the client promises that the aggregate traffic due to the related channels will remain within the client-specified bounds, and the network tries to use this information to reduce the resource allocation. An interesting issue is the interaction of this *globally specified* relationship with the admission control decisions that are made *locally* at the intermediate nodes (the local decision-making is due to the distributed nature of our resource sharing technique).

During channel establishment, the Real-Time Channel Administration Protocol (RCAP) [11] is responsible for admission control and resource reservation at each node. When the RCAP module at a certain node is reserving resources for a new channel, it allocates buffers for handling the delay jitter of the data packets from the previous node. This delay jitter depends on the resources allocated to the channel at the previous node (and the scheduling discipline followed at that node). Since channels that belong to the same sharing group may be routed to an intermediate node through different adjacent nodes, data from different member channels may arrive at a given node with different delay jitter. Thus, adding a new channel for an already established group may result in additional buffer allocation at that node.

6.2 Interactions with routing

The analysis and simulations described in [8] support the intuition that routing techniques affect the success of resource sharing. The routing algorithm can increase resource sharing gains in two ways:

- Routing of channels in the same sharing group along common subpaths.
- Routing channels so that they may have the same local delay bound along common subpaths.

The first effect is obvious, if the routing algorithm increases overlap between multicast trees for related channels, maximal resource sharing can be achieved. The second effect is the result of our decision (or the requirement of some scheduling algorithms) to give all channels in a sharing group the same local delay bound. If the routing algorithm is aware of the local delay bounds given to member channels, it can take these into account. Unfortunately, most algorithms do not enhance overlap of related multicast tree. In fact, adaptive routing algorithms may actually push channels of the same group away from each other to "less loaded" paths. The only way for the routing algorithm to enhance resource sharing performance is to recognize channels as members of sharing groups, and to favor shared subpaths between related channels. We have devised a routing algorithm that uses a heuristic to find a *min-cost* route that is likely to satisfy the delay requirements of all destinations. We have also adapted this algorithm to maintain state information for each sharing group, both to enhance path overlap and to meet the delay restrictions specified above.

6.3 Interactions with advance reservations

The Tenet research group has recently devised techniques for providing *advance reservations* for multi-party real-time communication [5]. In this service, the clients can make a request for network resources in advance of their use. A client may, for example, request a connection on Monday for a Wednesday video-conference. The network performs admission control tests and informs the client on the same Monday whether the desired resources will be available on Wednesday. Preliminary simulation results show a synergy between resource sharing and advance reservation mechanisms, i.e. advance reservations help increase resource sharing acceptance gain, under the assumption that larger conferences will be more likely to make reservations in advance.

7 Related work

A number of researchers have proposed techniques for providing *probabilistic* and *statistical* guarantees [18, 20] to improve the connection acceptance rates. Instead, resource sharing aims to improve the connection acceptance rate for *deterministic* guarantees by exploiting known relationships between channels. There are only two other related research efforts that we are aware of: the UCSD *filters* [17] and RSVP, the resource reservation protocol [21].

Pasquale et al.have proposed a stream *filter* that is "... an executable module which may be placed on a port, and implements a function which takes a specific set of streams associated with that port and produces a new stream" [17]. These filters perform an *application-level* transformation of one or more streams. A multiplexing filter could perform a function similar to the resource sharing mechanisms described in this paper by taking in streams from upstream nodes and multiplexing them on to a single output stream. It should be noted, however, that the design in this paper does not require application-level modules within the network. We are not aware of any implementation or any further research on these filters.

RSVP is a protocol by which receivers may reserve resources for *flows* (roughly equivalent to channels) that they receive. Along with reservation requests, receivers may specify a *reservation filter*, that describes which flows can use the reservation. The *Wildcard Filter* style corresponds most closely to the model of resource sharing described in this paper, in that the reserved resources may be used to forward data from any source in a given group. Other filtering styles are *Fixed* , in which the receiver specifies a list of enabled sources that *cannot be changed*, and *Dynamic*, in which receivers may dynamically alter the list of enabled sources.

The most important difference between the approach taken by RSVP and the resource sharing mechanisms described here for the Tenet scheme is that in RSVP the receiver determines the level of reservation, while in the Tenet scheme the reservation level is determined by the sender. Since many data streams cannot be scaled arbitrarily without serious degradation in perceived quality (e.g. [15]), we argue that the source should specify the resource requirements (at least in terms of bandwidth). Receiver control over bandwidth requirements can be obtained by using layered coding schemes [12] and putting each layer in a separate sharing group. This approach will be studied in future work.

8 Conclusions and suggestions for future work

We have presented a scheme for sharing resource allocations between guaranteed performance connections in computer networks. The scheme provides a fully-distributed, low-overhead technique for implementing resource sharing. We have evaluated the performance of this scheme by analysis and by simulations. Results [8] show that resource sharing is very useful in saving network resources. It achieves both higher connection acceptance rate and lower computational cost for admission control (than without resource sharing), while still providing guaranteed performance to the clients, *independent of the behavior of other, unrelated, traffic.*

We understand that routing affects the resource sharing benefits; we are currently exploring the relationship between routing and resource sharing. We expect that this research will help us design specific routing techniques that increase resource sharing benefits. We are also investigating the interactions between resource sharing and advance reservations. Also, a key component of real-time communication service is the traffic specification model which the client uses to characterize the channel traffic. A number of traffic models have been proposed in literature [1, 3, 14, 20]; we have to evaluate resource sharing benefits under different traffic specification models.

Acknowledgement

We want to thank Domenico Ferrari, who guided us in this research. Thanks are due to Riccardo Bettati, Andreas Hoffman, Bruce Mah, Steve Mccanne, Ron Widyono, Brian Whetten, Raj Yavatkar and Hui Zhang for their suggestions and comments on an earlier draft of this report.

This research was supported by the National Science Foundation and the Defense Advanced Research Projects Agency (DARPA) under Cooperative Agreement NCR-8919038 with the Corporation for National Research Initiatives, by AT&T Bell Laboratories, Hitachi, Ltd., Hitachi America, Ltd., Pacific Bell, the University of California under a MICRO grant, and the International Computer Science Institute. The views and conclusions contained in this document are those of the authors, and should not be interpreted as representing official policies, either expressed or implied, of the U.S. Government or any of the sponsoring organizations.

References

[1] Arthur W. Berger, Samuel P. Morgan, and Amy R. Reibman. Statistical multiplexing of layered video streams over ATM networks with leaky-bucket traffic descriptors, 1993. preprint.

[2] David Clark, Scott Shenker, and Lixia Zhang. Supporting real-time applications in an integrated services packet network: Architecture and mechanism. In *Proceedings of ACM SIGCOMM'92*, pages 14–26, Baltimore, Maryland, August 1992.

[3] Rene L. Cruz. A calculus for network delay, part I : Network elements in isolation. *IEEE Transaction of Information Theory*, 37(1):114–121, 1991.

[4] Domenico Ferrari, Anindo Banerjea, and Hui Zhang. Network support for multimedia: a discussion of the Tenet approach. Technical Report TR-92-072, International Computer Science Institute, Berkeley, California, October 1992. Also to appear in *Computer Networks and ISDN Systems*.

[5] Domenico Ferrari and Amit Gupta. Resource partitioning in real-time communication. In *Proceedings of IEEE Symposium on Global Data Networking*, Cairo, Egypt, December 1993.

[6] 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):368–379, April 1990.

[7] Amit Gupta, Wendy Heffner, Mark Moran, and Clemens Szyperski. Multi-party real-time communication in computer networks. In *Proceedings of 4th Interna-*

tional Workshop on Network and Operating Systems Support for Digital Audio and Video, Lancaster, UK, November 1993.

[8] Amit Gupta, Wingwai Howe, Quyen Nguyen, and Mark Moran. Evaluation of resource sharing beneffits. Technical Report TR-94-051, International Computer Science Institute, Berkeley, California, October 1994.

[9] Amit Gupta and Mark Moran. Channel groups: A unifying abstraction for specifying inter-stream relationships. Technical Report TR-93-015, International Computer Science Institute, Berkeley, California, March 1993.

[10] C.L. Liu and J.W. Layland. Scheduling algorithms for multiprogramming in a hard real-time environment. *Journal of ACM*, 20(1):46–61, January 1973.

[11] Bruce Mah. A mechanism for administration of real-time channels. Master's thesis, Tech. Report UCB/CSD-93-735, University of California, Berkeley, CA, March 1993.

[12] Steve McCanne, October 1994. Personal communication.

[13] Mark Moran and Riccardo Gusella. System support for efficient dynamically-configurable multi-party interactive multimedia applications. In *Proceedings of Thrid International Workshop on Network and Operating System Support for Digital Audio and Video*, San Diego, CA, November 1992.

[14] Antonio Ortega and Martin Vetterli. Multiple leaky buckets for increased statistical multiplexing of ATM video. In *Proceedngs of Packet Video Workshop*, Portland, OR, September 1994.

[15] Pramod Pancha and Magda El Zarki. A look at the MPEG video coding standard for variable bit rate video transmission. *Proc. IEEE INFOCOM '92*, May 1992.

[16] Colin Parris, Hui Zhang, and Domenico Ferrari. Dynamic management of guaranteed performance multimedia connections, April 1993. to appear in ACM Journal of Multimedia Systems.

[17] Joseph Pasquale, George Polyzos, Eric Anderson, and Vachaspati Kompella. The multimedia multicast channel. In *Proceedings of Third International Workshop on Network and Operating Systems Support for Distributed Audio and Video*, San Diego, CA, November 1992.

[18] Dinesh Verma. *Guaranteed Performance Communication in High Speed Networks*. PhD dissertation, University of California at Berkeley, November 1991.

[19] Hui Zhang and Domenico Ferrari. Rate-controlled static priority queueing. In *Proceedings of IEEE INFOCOM'93*, pages 227–236, San Francisco, California, April 1993.

[20] Hui Zhang and Ed Knightly. Providing end-to-end statistical performance guarantees with interval dependent stochastic models. In *ACM Sigmetrics'94*, Nashville, TN, May 1994.

[21] Lixia Zhang, Steve Deering, Deborah Estrin, Scott Shenker, and Daniel Zappala. RSVP: A new resource reservation protocol. *IEEE Networks Magazine*, 31(9):8–18, September 1993.