INTERNATIONAL COMPUTER SCIENCE INSTITUTE

1947 Center St. • Suite 600 • Berkeley, California 94704-1198 • (510) 643-9153 • FAX (510) 643-7684

Resource partitioning for multi-party real-time communication

Amit Gupta and Domenico Ferrari
{amit,ferrari}@tenet.berkeley.edu
Tenet Group
University of California at Berkeley, &
International Computer Science Institute

TR-94-061

November 1994

Abstract

For real-time communication services to achieve widespread usage, it is important that the network's management be allowed to control the services effectively. An important management capability concerns resource partitioning, i.e., distributing the different resources available at any given server (network node or link) among a number of partitions, where the admission control and establishment computations for a given connection need to consider only the connections in the same partition, and are completely independent of the connections accepted in other partitions. Resource partitioning is useful for a number of applications, including the creation of virtual private subnetworks, and of mechanisms for advance reservation of real-time network services, fast establishment of real-time connections, and mobile computing with real-time communication. In previous work, we presented a scheme for resource partitioning in a guaranteed performance networking environment with EDD-based packet scheduling disciplines. We now present the results of our continuing research, giving admission control tests for resource partitioning for two additional scheduling disciplines, FIFO and RCSP, as well. We also simulate our resource partitioning scheme in a multi-party application scenario. Our simulations confirm that resource fragmentation losses due to resource partitioning are small, and that resource partitioning reduces the admission control computation overhead. A somewhat surprising result from the simulation experiments is that, under circumstances that arise naturally in multi-party communication scenarios, resource partitioning results in *higher* overall connection acceptance rate.

1 Introduction

The increasing speed of computer networks and the improvement of workstation capabilities are enabling a new class of distributed applications, those involving multimedia data. It is widely believed that these applications should be supported in the general framework of real-time communication [1, 3, 13, 14, 15, 16], which provides predictable performance (e.g., end-to-end delay bounds for data delivery); typically, the clients negotiate with the network service provider to obtain a desired Quality of Service (QoS), which the provider guarantees. A number of schemes and protocols have been proposed to provide real-time communication services. Most of the schemes are connection-oriented, and reserve resources (bandwidth, buffers, and so on) along the route of a real-time connection [1, 3, 11]. QoS guarantees cannot be provided if the network does not check for saturation before accepting new channels. Thus, the network must perform *admission control* to ensure that the guarantees are met.

For real-time communication services to achieve widespread usage, it is important that the schemes and protocols allow the network's management to control the services effectively. A resource partitioning capability, i.e., the ability to distribute the different resources available at any network node or link among a number of partitions, is important for network managers. With resource partitioning, the admission control and establishment computations for a particular connection are independent of the connections accepted outside that connection's partition. This independence amounts to splitting the server into a number of sub-servers, each offering QoS guarantees only to the connections within it; the guarantees are valid as long as the admission tests and rate control schemes for all sub-servers are correct, and are independent of the individual per-connection establishment decisions and computations performed in other partitions.

This independence is very useful. The different sub-servers can be used to form virtual private subnetworks. The network's management can keep a small fraction of resources for management and fault-handling traffic. A fraction of the network's resources may be kept for non-real-time traffic. Other resource partitioning applications include advance reservations of resources for real-time channels [5], fast establishment of real-time connections, and support for mobile computing. The reader is referred to [4] for a more detailed description of these applications.

However, many important concerns remain regarding resource partitioning: for instance, whether we can design efficient mechanisms for resource reservation, and whether these mechanisms can be designed for different scheduling disciplines and admission control procedures. The efficiency concern encompasses two related issues. First, the computational expense associated with admission control should not appreciably increase under resource partitioning. Secondly, with every partitioning scheme, we have associated *resource fragmentation losses*. For example, consider a server with 100 resource units, with all requests being for 4 units each. This server can serve 25 such requests. On the other hand, if the server is partitioned into two sub-servers of 50 units capacity each, the two sub-servers would be able to serve only 24 such requests (and we would waste a total of 4 units of resources). For successful resource partitioning, the fragmentation losses should be kept to a negligibly low level.

Note that, as will be seen in Section 3, the derivation of some admission tests for a partition is nontrivial, due to the same reason that makes it hard to derive those admission tests in general: while testing against the still available amounts of bandwidth and buffer space (assuming static per-channel space allocation) in a server is easy if the required amounts of these two resources are known, verifying the schedulability of packets in a bounded-delay context after the addition of a new channel is much more complicated.

In a previous paper [4], we described a resource partitioning technique for networks with EDD-based packet scheduling disciplines. In this paper, we present the results of our continuing work, including resource partitioning techniques for two additional scheduling disciplines, FIFO and RCSP [17]. Thus, we show that our resource partitioning techniques are general, and apply to many scheduling disciplines. In the second part, we show that our techniques are useful and efficient; we report the results of some of our scheme's simulations, which show that the fragmentation loss is fairly small, and that resource partitioning appreciably *reduces* the computational overhead associated with admission control algorithms. In appropriate circumstances (which we will characterize more precisely in Section 4, and which occur naturally in multi-party real-time communication scenarios), resource partitioning can lead to higher overall connection acceptance rate. All simulation results presented here were obtained for EDD-based servers. The influence of the specific scheduling policy on the conclusions one can draw from the experiments is so negligible that we chose to show results for only one such policy.

The paper is organized as follows. Section 2 provides the background for the research described here. In Section 3 we discuss the admission control tests for an EDD-based, a FIFO-based, and an RCSP-based packet scheduler with support for resource partitioning. Section 4 presents a simulation-based evaluation of resource partitioning algorithms in a multi-party communication environment. We discuss implementation issues in Section 5, and conclude this paper with a brief review of related work in Section 6.

2 Background

In this paper, we illustrate our approach to network resource partitioning in the framework of the Tenet real-time communication protocols [3]. Our previous paper [4] describes the relevant aspects of the Tenet protocols; in this discussion, we limit ourselves to discussing a few of their salient features. For simplicity, we also limit our description to unicast connections, though our resource partitioning techniques apply also to a multicast environment such as the one supported by the new generation of Tenet protocols now being developed [8].

The Tenet protocols are based on a network-layer abstraction, the *real-time channel*, characterized by traffic and performance bounds specified by the client. The traffic bounds may be described in terms of a number of different traffic specification models, including leaky bucket, $Xmin$-$Xave$-I [6], and so on. The client also specifies the desired quality of service in terms of bounds on several performance indices: these indices include end-to-end delay, delay jitter (i.e., variation), loss rate, and the probabilities that these bounds will be met.

The Tenet approach is connection-oriented and reservation-based: before a real-time channel can be used by its requester, it must be established (i.e., resources for the channel must be set aside along its route), so that the desired performance guarantees can be provided. Note that these resources are not reserved for the channel's exclusive use; the channel only has higher priority for using them, but, in keeping with the principles of packet switching, they can be used by non-real-time traffic whenever no packets from the channel are present in a node or on a link.

Channel establishment is a distributed process. A message issued by the source (though it could be issued by the destination instead) visits each node (switch, router, gateway) on the route of the channel. This message causes several admission tests and computations to be performed at each node. If the new channel passes all the tests in a node, the message is forwarded, with some state information about the current node, to the next node on the route. The final tests are performed by the destination; if they are successful, a channel-established message is sent by the destination to the source along the reverse route; when each node is revisited, the message corrects the tentative reservations made in that node by the forward message, and informs the node about the performance bounds assigned to it. These revised bounds are used later during the data transfer operations, and for the admission of future channels.

If any of the tests in the nodes or in the destination fails, the channel cannot be established, and a channel-rejected message is immediately sent back to the source. This message removes in each node it visits the tentative reservations made for the new channel.

To complete this general description, we only have to mention two special aspects of data transfers: scheduling and distributed rate-control. Most scheduling policies can be used for real-time communication under fairly liberal conditions, and can co-exist within the same network and along the same channel [2]; our partitioning scheme applies to such a mixed scheduling situation, as long as the appropriate admission tests are used in each server. Also, rate control, either at the periphery of the network or in all of its nodes, is needed to protect well-behaving channels from the misbehavior of faulty or malicious sources, and from the occasional bunching of packets on a channel due to traffic fluctuations. In our previous work [4], we described the admission control tests for EDD-based scheduling disciplines. In the next section, we re-state these tests, and also present partition-based admission control tests for two more scheduling disciplines, FIFO and the new Rate-Controlled Static Priority (RCSP) policy [17].

3 Resource partitioning tests

What parts of an admission control algorithm do we need to modify to make network resources partitionable? Our ideal objective is to subdivide a network (or, more generally, an internetwork) into a certain number of virtual networks (or internetworks), each one of which can be treated totally independently of the others, even though they all share the same hosts, nodes, and links. This is equivalent to saying that we would like to confine our admission tests to the resources assigned to the partition to which a new channel is requesting admission, without in any way involving the channels and the resources belonging to the other partitions. We previously showed that this goal can be reached for EDD-based packet scheduling disciplines [4]. In this section, we also present admission tests for two more scheduling disciplines, FIFO and RCSP, and describe the simple changes to be made to the respective admission control tests for supporting resource partitioning.

Note that, to be accepted, a request for a new channel must pass also a buffer space test in each server [4]. This test is very easy to derive, and will be omitted for the sake of brevity throughout our discussion.

3.1 Resource partitioning in an EDD server

In this section, we describe the admission control tests for an EDD server, with and without resource partitioning. We will limit this description to stating the admission control tests; the interested reader can find the proofs for these admission control tests for EDD servers with and without resource partitioning in [4] and [6] respectively.

In an EDD-scheduled server, an established channel k is characterized by t_k, the maximum service time for any packet belonging to this channel, and d_k, the local delay bound (which is the maximum amount of time that any packet on this channel will stay in this server). In addition, the server itself is characterized by t^*, the maximum service time for any packet (real-time or best effort) serviced. We assume, for brevity of explanation, that the admission control tests maintain the list of already established channels as sorted in non-decreasing order of local delay bounds ($d_i \geq d_j$ if $i \geq j$). For a new resource request R with maximum packet service time t_{new}, we can assign it a delay bound d_{new}, causing the new channel to be inserted without causing any of the local delay bounds to be violated if, after adding this new connection,

$$d_h \geq \sum_{l=1}^{h} t_l + t^*. \tag{1}$$

where the index h goes over all real-time channels in that server.

The test ensures that, when a packet arrives for a channel k, the maximum amount of time it could possibly wait (before being transmitted) is bounded above by the delay bound d_k.

We now state the admission control tests for an EDD server with resource partitioning. Consider an EDD server with partitions $P_1, ...P_n$, partition P_s being allocated a fraction a_s of the server resources such that $\sum_{s=1}^{n} a_s \leq 1$. The EDD server will guarantee this delay bound to all packets of partition P_s (with allocation a_s) if

$$d_h \geq \frac{1}{a_s} \sum_{l=1}^{h} t_l + t^*, \tag{2}$$

where the index h goes over all channels in partition P_s.

Note that, for EDD servers, there is also a bandwidth test, whose adoption to the partitioning case is, however, trivial and will not be described here. The interested reader can find it in [4].

3.2 Resource partitioning in a FIFO server

We now describe the admission control test for a FIFO server without resource partitioning. In a FIFO server, all real-time connections are assigned the same local delay bound, say d. Let traffic over a real-time connection be characterized at the network layer by the quadruple $(Xmin, Xave, I, Smax)$, where $Xmin$ is the minimum interpacket interval, $Xave$ is the minimum average interpacket interval, I is the averaging interval, and $Smax$ is the maximum packet size. For a new resource request R with traffic specification $(Xmin, Xave, I, Smax)$, we can assign local delay bound d to this new connection without violating the delay bounds of existing connections if, after adding this new connection,

$$\sum_i \lceil \frac{d}{Xmin_i} \rceil * Smax_i + Smax^* \leq d * ServiceRate. \tag{3}$$

where the index i goes over all real-time channels at that server, and $Smax^*$ is the size of the largest packet (either real-time or best-effort) that is to be serviced by this server.

The test ensures that, when a packet arrives, the maximum amount of time it could possibly wait (before being transmitted) is bounded above by the delay bound d associated with that server. It is easy to see that

$$0 < Smax^* \leq d * ServiceRate. \tag{4}$$

We now introduce the FIFO admission control tests with resource partitioning.

Theorem 1 *Consider a FIFO server with delay bound d, and partitions $P_1, ...P_n$ with partition P_s allocated a fraction a_s of the server resources such that*

$$\sum_{s=1}^{n} a_s \leq 1. \tag{5}$$

The FIFO server will guarantee this delay bound to all packets of partition P_s (with allocation a_s) if

$$\sum_i \lceil \frac{d}{Xmin_i} \rceil * Smax_i + Smax^* * a_s \leq d * ServiceRate * a_s, \quad (6)$$

where the index i goes over all channels in partition P_s.

Proof We say that the test in (6) is valid if it only admits channels to the given partition that always satisfy the test in (3) applied to the entire population of channels. The validity of (6) can be observed by adding the admission control tests over all partitions to obtain

$$\sum_i \lceil \frac{d}{Xmin_i} \rceil * Smax_i + Smax^* * \sum_{s=1}^{n} a_s \leq d * ServiceRate * \sum_{s=1}^{n} a_s. \quad (7)$$

where the index i goes over all connections in the FIFO server. Substituting (4) and (5) in (7), we obtain

$$\sum_i \lceil \frac{d}{Xmin_i} \rceil * Smax_i + Smax^* \leq d * ServiceRate. \quad (8)$$

that is, the resource-partitioning test in (3). **Q.E.D.**

3.3 Resource partitioning in an RCSP server

3.3.1 A brief introduction to RCSP

The interested reader can find a detailed description of RCSP in [17]; here, we present a simplified description of the discipline. Figure 1 shows an RCSP server for a node with a single output link; additional links can be added by replicating the scheduler portion of the server. Only the handling of real-time traffic is shown in the figure; non-real-time traffic is collected from the input links into per-output-link queues, each of which has the lowest static priority among the queues for the corresponding outgoing link.

An RCSP server has two components: a rate controller and a static-priority scheduler. The rate controller shapes the input traffic from each connection (so that the packets do not violate the traffic specification when they go into the scheduler); the scheduler orders the transmissions of the packets from all connections. By neatly separating the rate-control and delay-control functions in this manner, RCSP achieves flexibility in allocation of delay and bandwidth, as well as simplicity of implementation.

Conceptually, a rate controller consists of a set of regulators corresponding to each of the connections traversing the switch; each regulator is responsible for shaping the input traffic of the corresponding connection into the desired traffic pattern.

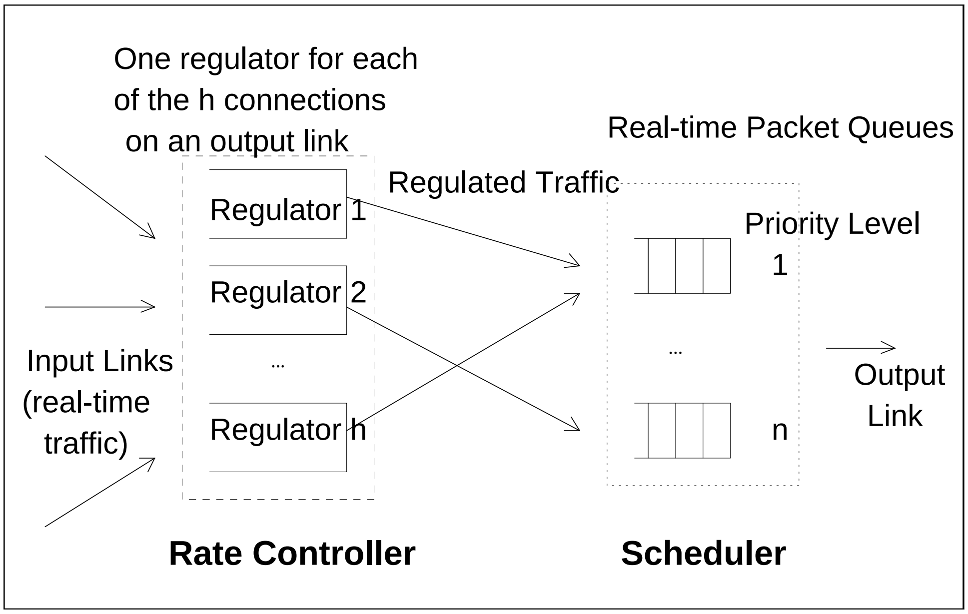

Figure 1: An RCSP server (from [17])

Regulators control the interactions between switches and reduce or eliminate jitter. Regulators achieve this control by holding data packets for the appropriate amount of time before handing them to the scheduler.

The scheduler services packets using a non-preemptive static-priority discipline:

- when the server chooses the next packet to transmit, the packet at the head of the highest-priority non-empty real-time queue is chosen; the packets in each real-time queue are serviced on a first-come-first-served basis;
- non-real-time packets are transmitted only when there are no real-time packets in the scheduler;
- the transmission of a lower-priority packet is not preempted by the arrival of a higher-priority packet.

3.3.2 RCSP admission control tests

Network resources can be partitioned if, given a number of partitions and suitable admission tests to each partition, we can prove that all channels established in all partitions always satisfy the admission tests for the full network.

We first describe the admission control tests for RCSP servers without resource partitioning. Consider an RCSP scheduler with P priority levels, and with D_i as the delay bound associated with priority level i. Let traffic over a real-time connection be characterized at the network layer by the quadruple $(Xmin, Xave, I, Smax)$, where $Xmin$ is the minimum interpacket interval, $Xave$ is the minimum average interpacket interval, I is the averaging interval, and $Smax$ is the maximum packet size. For a new resource request R with traffic specification $(Xmin, Xave, I, Smax)$, we can assign a local delay bound D_m (the delay bound associated with priority level m) to this new connection without violating the delay bounds of existing connections if, after adding this new connection, for all priority levels l, $1 \leq l \leq P$,

$$\sum_i \lceil \frac{D_l}{Xmin_i} \rceil * Smax_i + Smax^* \leq D_l * ServiceRate. \tag{9}$$

where the index i goes over all channels at or above the priority level l, and $Smax^*$ is the largest packet size that can be serviced by this server.

Intuitively, the tests ensure that, when a packet arrives (for a connection at priority level p), the maximum amount of time it could possibly wait (before being transmitted) is bounded above by the delay bound D_p associated with that priority level. We will not rigorously prove this RCSP admission control tests here; the interested reader can find the proof in [17]. In practice, the terms $\sum_i \lceil \frac{D_l}{Xmin_i} \rceil * Smax_i + Smax^*$ can be incrementally re-computed each time a new connection is accepted or torn down. This leads to fast, $O(P)$ admission control tests for RCSP, where P is the number of priority levels.

We now introduce the admission control tests for RCSP servers with resource partitioning. Consider an RCSP scheduler with P priority levels, a partition with a fraction f of the server's resources, and with D_i as the delay bound associated with priority level i. Note that, for the sake of simplicity, we assume that all resource types in a server are partitioned in the same proportions, though the value of f for a given partition can still change from server to server. We also assume, for simplicity, that the same fraction f that characterizes a partition in a server applies to all P priority levels. The RCSP server will guarantee these delay bounds to all packets of this partition if, for all priority levels l, $1 \leq l \leq P$,

$$\sum_i \lceil \frac{D_l}{Xmin_i} \rceil * Smax_i + Smax^* * f \leq D_l * ServiceRate * f. \tag{10}$$

where the index i goes over all channels in that partition, as long as the sum of resource allocations to all partitions does not exceed unity.

We say that the tests (10) are valid if they only admit channels to the given partitions that always satisfy tests (9) applied to the entire population of channels. We now prove the following assertion.

Theorem 2 *Consider a server with partitions $P_1, ...P_n$, and allocation a_s for partition P_s. Under the condition:*

$$\sum_{s=1}^{n} a_s \leq 1 \tag{11}$$

the resource partitioning tests described in (10) are valid.

Proof From the resource partitioning tests (10) above, we know that, for all levels $t, 1 \leq t \leq l$, for all partitions $P_s, 1 \leq s \leq n$, the following condition holds:

$$\sum_i \lceil \frac{D_l}{Xmin_i} \rceil * Smax_i + Smax^* * a_s \leq D_l * ServiceRate * a_s, \tag{12}$$

where the index i goes over all channels in partition P_s at or above the priority level t, and $Smax^*$ is the largest packet size that is to be serviced by this server.

At level t, we add up tests (10) for all partitions, and we obtain

$$\sum_i \lceil \frac{D_l}{Xmin_i} \rceil * Smax_i + Smax^* * \sum_{s=1}^{n} a_s \leq D_l * ServiceRate * \sum_{s=1}^{n} a_s \tag{13}$$

where the index i goes over all channels in all partitions at that server at or above the priority level t, and $Smax^*$ is again the largest packet size that is to be serviced by this server.

Since we have

$$Smax^* \leq D_l * ServiceRate, \tag{14}$$

for all levels $l, 1 \leq l \leq P$, substituting (14) and (11) into (13), we obtain

$$\sum_i \lceil \frac{D_l}{Xmin_i} \rceil * Smax_i + Smax^* \leq D_l * ServiceRate, \tag{15}$$

that is, the resource partitioning tests in (10). **Q.E.D.**

4 Simulations

In the previous section, we demonstrated the generality of our resource partitioning techniques – they can be applied to many scheduling disciplines. In this section, we show that these techniques are useful and efficient.

We performed a number of simulation experiments to evaluate the performance of the resource partitioning algorithms. For this discussion, we have selected two interesting sets of simulation experiments. In the first set, we ran simulations of resource requests for simplex unicast connections. In the second set, we ran simulations of resource requests for multi-party communication, including distributed video-conferences. In this section, we describe the simulation scenario, the workload, and the results obtained with these experiments. We ran these simulations on *Galileo* [10], an object-oriented real-time network simulator.

Our goal was to make the experiments as realistic as possible, so that we could confidently predict the behavior of our implementation of resource partitioning in the Tenet Protocol Suite 2 [8]. For example, we used the NSFNET backbone network topology in our simulations (see Figure 2). We assumed that the rate of each link was 45 Mbps, the propagation delay along the diameter was 40 ms, and that we could allocate up to 80% of the resources for real-time communication. We also made the

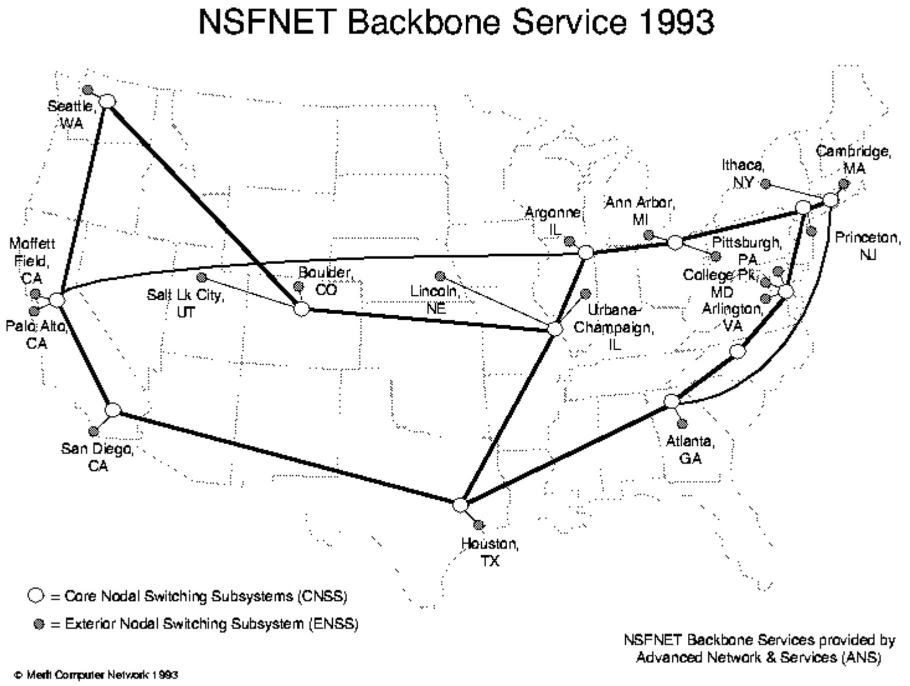

Figure 2: The NSFNET topology

amount of buffer space in each server large enough that the buffer space test would always be successful. Thus, bandwidth or processing power was the limiting resource in all servers and all scenarios.

Simulation workload and evaluation metrics

In all the experiments, the sources and destinations for the channels were uniformly and independently distributed among the network nodes. We ran the same simulations without resource partitioning and with resource partitioning for different resource allocations to the partitions.

To keep comparisons meaningful, we only considered relatively homogeneous workloads, where all channels have identical traffic descriptions, and all destinations specify identical performance requirements:

- Deterministic delay bound $D = 400$ ms ;
- Deterministic jitter bound $J = 16$ ms;
- Minimum inter-packet time $Xmin = 8$ ms;
- Maximum packet size = 8 Kb.

This corresponds to a compressed video stream at a peak rate of 1 Mbps, at 30 frames-per-second with four data packets per frame ; the average rate did not matter, because the admission control tests in this simulation used peak-rate bandwidth allocation.

We performed many sets of experiments with varying workload parameters; for each of these workloads, we ran many simulation experiments, and averaged the results thus obtained.

The main metric we adopted for evaluation and comparison was the *acceptance ratio*:

$$\text{Acceptance ratio} = \frac{\text{Number of destinations reached with resource partitioning}}{\text{Number of destinations reached without resource partitioning}}$$

We were also interested in the timeliness and the computational cost of channel establishment, for which we used a different metric: the computational overhead associated with admission control. In the simulations, we used the EDD scheduling discipline [6], in which admission control tests at a node take $O(n)$ time, where n is the number of resource allocations (where an allocation is a resource reservation for a channel, or for a group of channels) existing at that server. We summed the values of n (the number of existing channels when a new request arrives at a node) at *all* nodes on the new channel's path to obtain a *"number of steps"* measure for the total computational overhead. This led us to the following metric for comparing computational overhead

$$\text{Overhead ratio} = \frac{\text{Total computational overhead with resource partitioning}}{\text{Total computational overhead without resource partitioning}}$$

4.1 Homogeneous requests

In the first set of experiments, we ran our simulations with simplex unicast connections alone; with these connections, quantitative comparisons using the metrics described above are particularly easy to make.

We compared the following two scenarios:

- 300 simplex unicast connections, all in the same partition, which is allocated 80% of the network's bandwidth; we call this the *"without resource partitioning"* case; and

- two partitions with 150 simplex unicast connections each, and varying partition allocations, so that the total resource allocation for these partitions equals 80% of the network's bandwidth; this is the *"with resource partitioning"* case.

We deliberately chose this workload to saturate the network, because we wanted to observe the network's behavior under heavy real-time load.

Results

In Figure 3 and all of the remaining figures, we report on the horizontal axis the fraction f of the total resources that is allocated to one of the partitions. The other partition's allocation is $100(0.8 - f)\%$ of the total resources.

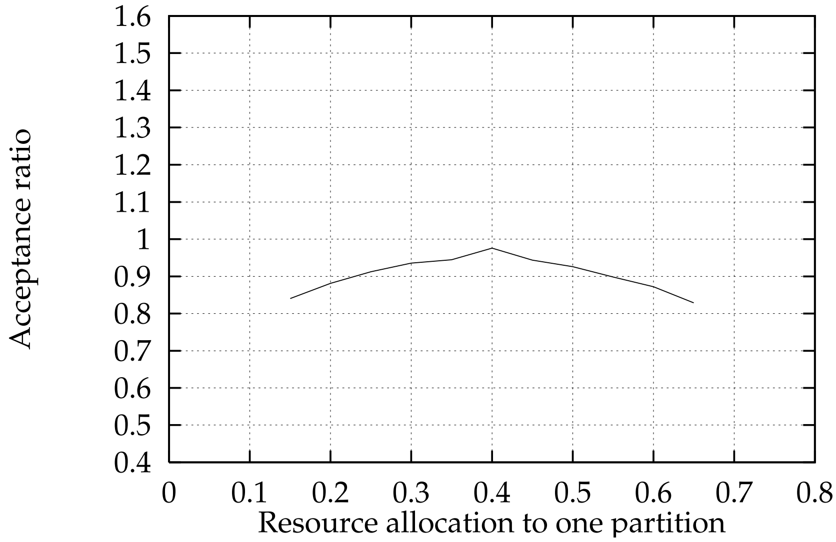

Figure 3: Acceptance ratio vs. relative partition allocation for unicast channels

- **Acceptance ratio** As we mentioned in Section 1, with resource partitioning we can expect fragmentation losses; in Figure 3, we observe fragmentation losses of up to 20%, depending on the relative resource allocations to the partitions. This graph also verifies that the partitioning scheme works fairly well, since, if the partition allocations are appropriately chosen, the fragmentation losses are fairly minimal (about 2-3%). As all channels are identical and equally distributed between the two partitions, the best choice is that of identical allocations. Note that this curve is, as expected, symmetrical with respect to a 0.4 relative partition allocation.

- **Computational overhead** Resource partitioning reduces the computational overhead associated with admission control because, during admission control tests for a new connection, we only have to consider other connections of the same partition; without resource partitioning, we would have to consider all connections at that server. In this experiment, we expect resource partitioning to lead to a reduction of about 50% in computational overhead, and the graph in Figure 4 verifies this intuition.

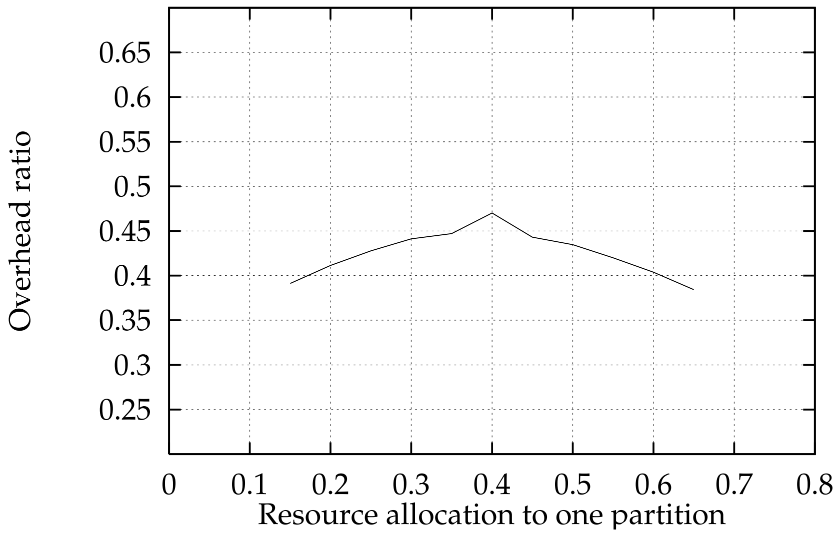

Figure 4: Computational overhead vs. relative partition allocation for unicast channels

4.2 Multi-party communication

In the second set of experiments, we considered heterogeneous requests. Here, some requests were for simplex unicast connections; the others were for conferences, where the participants could *share* resources [9]. We considered the case where the conference requests were all served by one partition, while the unicast connection requests were served by the other partition. This segregation is a natural consequence of some aspects of multi-party communication, for instance advance reservation requirements [5].

We compared the following scenarios:

- 300 simplex unicast connections, all in the same partition, which was allocated 80% of the network's bandwidth; as above, this is called the *without resource partitioning* case; and

- two partitions, with 150 simplex unicast connections in the first partition, and 50 10-person conferences in the second partition, and with varying partition allocations, so that the total resource allocation for these partitions equals 80% of the network's bandwidth.

Results

- **Acceptance ratio** The graph in Figure 5 shows an interesting result. There is a fairly large region in which the overall connection acceptance ratio is higher than one, i.e. the acceptance rate is higher with resource partitioning (than

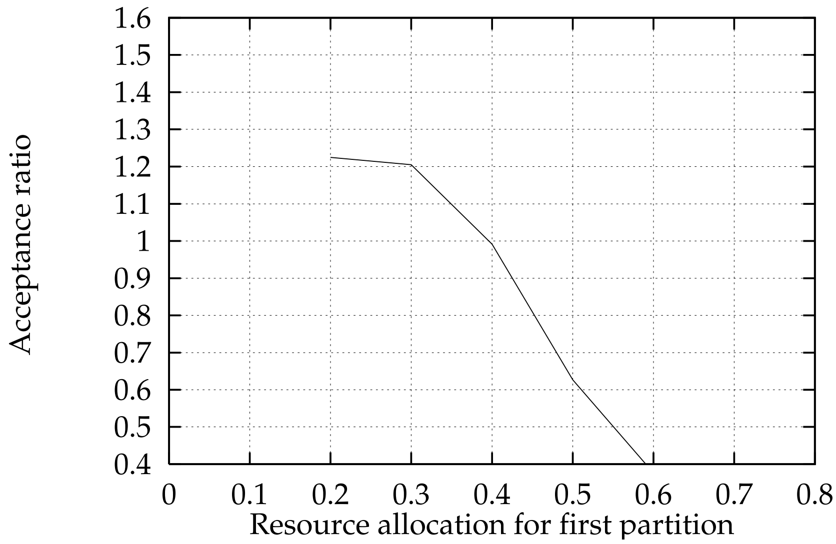

Figure 5: Acceptance ratio vs. relative partition allocation for the multi-party communication scenario. The first partition is for unicast connections.

without resource partitioning). This phenomenon is easily explained in the following manner. First, resource allocation requests have varying efficiencies in using resources. In particular, with resource sharing, the resource requirements do not increase with allocation requests for additional channels [9]; this implies that conference requests are more efficient in using resources than isolated connections. Second, as we mentioned before, partitioning provides protection for allocation among partitions; in this case, partitioning ensures that the resources allocated to the first partition will only be used for conferences, and not for isolated connections. As the conferences use resources more efficiently, the acceptance gains with conferences may be large enough to offset the fragmentation losses that we observed in the previous experiment.

Resource partitioning leads to higher connection acceptance ratios when the network is overloaded (i.e., when the total request for resources exceeds the supply, and thus, the admission control has to refuse some requests) and when the partition(s) that accommodate resource-sharing requests do not service a large number of isolated connections. This segregation of requests (conferences vs. isolated connections) happens naturally in multi-party communication environments where the conferences are scheduled well in advance, and the advance reservation requests use a separate partition [5].

- **Computational overhead** In the graph in Figure 6, we observe again that resource partitioning reduces the overall computational overhead in the admission control process.

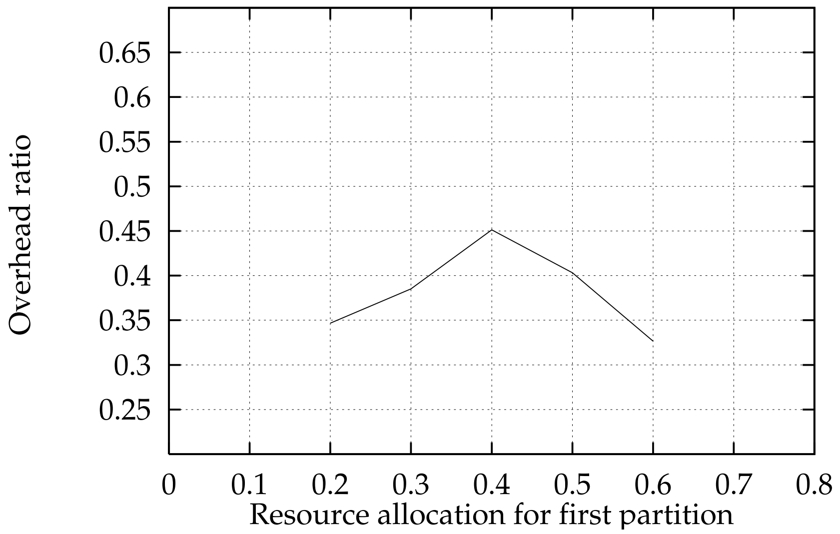

Figure 6: Computational overhead vs. relative partition allocation for the multi-party communication scenario.

5 Implementation

In the Tenet protocol suites, the Real-time Channel Administration Protocol (RCAP) [12] performs resource reservation and admission control for real-time connections. Resource partitioning affects the following aspects of RCAP:

- the admission control tests;
- the client-service interface (how does a client specify the partition(s) a resource reservation request may use?);
- the network configuration management functions (how do we set up partitions on the network nodes? how do we set up (and change) the resource allocations for the partitions?);
- the interactions with other components, including routing, resource sharing, and authorization and access control.

The Tenet protocols are designed to support a number of scheduling disciplines. However, for the initial implementation of Suite 2, we adopted the RCSP discipline, and use the admission control tests described in Section 3.3.2.

RCAP exports two different interfaces: one to the network service clients, the other to the network service managers. The client interface is very simple; with every connection request, the client specifies the partition whose resources that connection should use.

When RCAP starts up at a node, by default, it creates one partition containing all node resources. RCAP offers the following partition manipulation primitives to the network managers:

- CreatePartition(): To create a new partition on that node.
- DeletePartition() : To delete an existing partition on that node.
- GetAllocation(): To obtain the current resource allocations for a given partition.
- SetAllocation(): To set new resource allocations for a given partition.
- TestAllocation(): To test if setting new resource allocations may violate the guarantees given to existing connections.

Resource partitioning also affects many other aspects, including routing, resource sharing, and access control. By resource partitioning, we create multiple independent subnetworks; this affects routing in two manners: first, the routing service should treat different partitions independently; second, changing partition allocation amounts to changing the characteristics of the corresponding servers, and the routing service must update its tables accordingly.

What do we do when we learn that two (or more) channels can share resources, though they belong to separate partitions? We have taken the conservative approach of limiting the scope of resource sharing to connections within a resource partition. In the future, we would like to investigate some other approaches, including moving channels between partitions, or having more than one partition contribute resources for a given connection.

Resource partitioning introduces several access control issues. First, how do clients obtain permission to create a channel in a particular partition, and how does the network verify that these clients have the desired authorization? Second, how does the network prevent unauthorized access to the partition manipulation primitives described above? In the current prototype implementation, we have a fairly simplistic capability-based access control; unfortunately, this control does not address a number of authentication issues. We are now designing public-key-encryption based techniques for access control, which we will incorporate later in Suite 2.

6 Related Work

To our knowledge, only one other scheme, the *CBQ* scheme, has been proposed for network resource partitioning [7]. The primary motivation of *CBQ* is to provide a service in which the virtual subnetworks are used by the network's managers to control and equitably distribute the available bandwidth among different organizations. In the *CBQ* scheme, the network observes the data rates for different traffic classes, and, when the load increases (the queues build up), the network provides some form of fair queuing to ensure that the different partitions use the bandwidth of each link in proportion to their allocations.

The *CBQ* scheme has been designed for bandwidth distribution to non-real-time traffic (and for non-guaranteed schemes like *predicted service* [1]). *CBQ* modifies the packet scheduling policy to manage and distribute the short-term resource usage. In its current form, it cannot provide partitioning for guaranteed-performance communication. Our scheme is designed for real-time traffic, implements resource partitioning at channel establishment time, and does not require any changes in per-packet scheduling. The two schemes can co-exist, with the *CBQ* scheme administering resources for non-real-time traffic.

7 Conclusions

We have devised techniques for resource partitioning in real-time networks, where the partitioning computations are limited to channel establishment time; per-packet scheduling and data forwarding are not affected by partitions. Our resource partitioning techniques are general – they apply to many scheduling disciplines. We presented partition-based admission control algorithms for EDD-based, FIFO-based, and RCSP-based packet schedulers. We also showed that our techniques are useful and efficient. Our simulations show that resource partitioning can substantially reduce the computational overhead associated with admission control for real-time connections. Also, under the circumstances described in Section 4, resource partitioning techniques may result in higher overall connection acceptance ratios.

We have provided resource partitioning techniques for many scheduling disciplines; however, for useful resource partitioning, we need to solve a few additional problems. Of great importance are access control and, more generally, security. We have to design techniques and mechanisms to decide if a resource reservation request has the privileges needed to obtain resources from the requested partition(s). We also have to authenticate the network control messages that manipulate resource partition allocations.

Acknowledgment

The authors are grateful to Hui Zhang, who provided the RCSP server diagram in Figure 1. We would also like to thank Anindo Banerjea, Bruce Mah, and Hui Zhang for many useful comments on a previous draft of this paper. 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, Digital Equipment Corporation, Hitachi, Ltd., Pacific Bell, 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] 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.

[2] Domenico Ferrari. Real-time communication in an internetwork. *Journal of High Speed Networks*, 1(1):79–103, 1992.

[3] Domenico Ferrari, Anindo Banerjea, and Hui Zhang. Network support for multimedia: a discussion of the Tenet approach. *Computer Networks and ISDN Systems*, pages 1267–1280, July 1994.

[4] Domenico Ferrari and Amit Gupta. Resource partitioning in real-time communication. In *Proceedings of IEEE Symposium on Global Data Networking*, pages 128–135, Cairo, Egypt, December 1993.

[5] Domenico Ferrari, Amit Gupta, and Giorgio Ventre. Advance reservations for real-time communication. Unpublished, 1994.

[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] Sally Floyd. Link-sharing and resource management models for packet networks. Unpublished, September 1993.

[8] Amit Gupta, Wendy Heffner, Mark Moran, and Clemens Szyperski. Multi-party real-time communication in computer networks. In *Collected abstracts of 4th International Workshop on Network and Operating Systems Support for Digital Audio and Video*, pages 37–39, Lancaster, UK, November 1993.

[9] Amit Gupta, Winnie Howe, Mark Moran, and Quyen Nguyen. Resource sharing in multi-party realtime communication. In *To appear in Proceedings of INFOCOM 95*, Boston, MA, April 1995.

[10] E. W. Knightly and G. Ventre. Galileo: a tool for simulation and analysis of real-time networks. In *Proceedings of IEEE 1993 International Conference on Network Protocols*, pages 264–271, San Francisco, CA, October 1993.

[11] A. Lazar and C. Pacifici. Control of resources in broadband networks with quality of service guarantees. *IEEE Communication Magazine*, pages 66–73, October 1991.

[12] 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.

[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*, pages 143–156, San Diego, CA, November 1992.

[14] Abhay Kumar J. Parekh. *A Generalized Processor Sharing Approach to Flow Control in Integrated Services Networks.* PhD dissertation, Massachusetts Institute of Technology, February 1992.

[15] Craig Partridge and Stephen Pink. An implementation of the revised internet stream protocol (ST-2). In *Journal of Internetworking Research and Experience*, pages 27–54, 1992.

[16] Jean Ramaekers and Giorgio Ventre. Client-network interaction in a real-time communication environment. In *Proceedings of GLOBECOMM '92*, pages 1140–1144, Orlando, Florida, December 1992.

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