A Three-Pass Establishment Protocol for Real-Time Multiparty Communication

Jaehee Yoon Riccardo Bettati

Department of Computer Science

Texas A&M University

College Station, TX, 77843-3112

E-mail: {jaeheey,bettati}@cs.tamu.edu

Technical Report 97-006

July 14, 1997

Abstract

The development of high-speed networks and global internetworking protocols enable new, multimedia-oriented, applications to emerge, such as teleconferencing and other collaborative applications, or video-on-demand. These new applications rely on the underlying communication infrastructure to be able to provide Quality of Service (QoS) guarantees, typically in form of delay and jitter bounds. Appropriate admission control during connection establishment is necessary for the underlying communication infrastructure to satisfy the above QoS requirements. Admission control is used in conjunction with a resource reservation protocol that reserve resources at communication establishment time. Existing resource reservation algorithms are limited when it comes to reserving resources for multiparty communication environments, in particular for multicast connections. We propose a new protocol for connection establishment in a real-time one-to-many communication environment. This algorithm very effectively allocates resources for connections along their multicast routes, and thus reduces call-blocking probabilities and increases resource utilization while providing QoS guarantees. This is illustrated with a suite of simulation experiments, which compares the performance of our approach with that of other existing protocols.

1 Introduction

Many sophisticated emerging applications, such as distributed interactive simulation, multimedia conferencing, or distributed virtual reality, are difficult to deploy in todays internetworking infrastructure. This is mainly due to one requirement that all these applications share: the need for guaranteed real-time service. These applications not only require high bandwidth, but predictable *Quality of Service (QoS)* as well.

The QoS requirements at network level are typically specified in terms of bounds on worst-case end-to-end delay and on the worst-case delay jitter for packets of the connection. Other parameters may be specified as well, such as packet loss rate, deadline miss rate, but we will not further address them in this discussion. The desired delivery time for each message across the network is bounded by a *deadline*, a specific maximum delivery delay. This delay bound is an application-layer, end-to-end timing constraint. If a message arrives after the deadline is expired, the message is useless and is typically discarded. For most real-time applications, it is not important how fast a message is delivered. Indeed, packets arriving early may need to be buffered at the receiver to achieve constant end-to-end delay. Therefore, *delay jitter*, which is the maximum variation in delay experienced by packets in a single connection, is another performance metric. For example, in video transmission, jitter may cause some frames to arrive early, and others to arrive late. Although the transmission of all frames satisfies the deadline requirement, the displayed movie may appear jittery. *Buffers* at the receiver can be used to control delay jitter. The amount of buffer space required can be determined from the peak rate and the delay jitter of the delivery process and can be quite large for a network with no control of delay.

Real-time communication services require clients to specify traffic parameters of connections along with their QoS requirements (also called performance requirements) to obtain guarantees. QoS requirements are typically defined by constraints imposed by the application environment, e.g., frame rates for video transmission, interactivity thresholds for audio conferencing, or stability requirements for control applications. During connection establishment, the underlying network service is informed about the QoS requirements, typically as part of a connection-request message. If the connection satisfies the requirements without endangering guarantees given to existing connections, the connection is established for the client. At this point, the client and the network service enter a "real-time service contract": as long as the client adheres to the traffic specification (and the network service can enforce this by *policing*), the network service must guarantee the agreed-upon QoS requirements.

1.1 Multiparty Communication

An increasing number of applications require support for multipary communication at network-service level. For example, teleseminar applications require one-to-many connections. Teleconferencing, on the other hand, requires a one-to-many or many-to-many network delivery service. Distributed interactive simulation systems need support for multicast receiver groups with potentially large and dynamically changing memberschip. Publish/subscribe methods to system-level fault tolerance may not involve large memberships, but take advantage of the separation of senders from receivers typical for multiparty communication schemes.

In order to conserve network resources, support for multiparty application requires the capability to do *multicasting*. Multicasting permits sources to send a single copy of a message to an address that causes the message to be delivered to multiple recipients. Multicasting is far more efficient than sending separate copies of the same message to each recipient, each over a separate unicast connection. In addition to the obvious reduction in wasted bandwidth the sender is released from having to manage the set of receivers.

Adding multicasting capabilities to internetworks, has been achieved only in the past few years, primarily with multicast IP [6]. Multicast IP consists of extensions to routing protocols to allow subnetworks to advertise (through their routers) whether they wish to receive datagrams sent to particular sets of servers (called multicast group). Routing information for a particular multicast group is shared. There is no need to set up explicit delivery trees for each sender. The trees are implicitly computed when multicast datagrams arrive, and if trees from two different sources overlap, the same routing information can be used.

1.2 Resource Allocation Protocols

In order to associate traffic with its QoS requirements, protocols with support for QoS guarantees are typically connection oriented. An important step in connection establishment in such protocols is the *admission control*. The purpose of admission control is to calculate which network resources, such as buffer, bandwidth, and priority queues, are required in order to provide the QoS guarantees, and to determine if enough such resources are available. If this is the case, connection establishment reserves these resources for the new connection. When a client requests sending data from a source node to some destinations, an *establishment message* is typically sent out by the source, carrying the traffic parameters of the requested connection. This message follows the route of the connection, triggering an *admission test* in each node it traverses, and tentatively reserving resources in the node if all tests are passed. The resource reservation for the new connection must not endanger the guarantees given to the already established connections, so the admission test checks this condition with a local resource manager in each node. If the test fails, the connection request is denied. The establishment process is aborted by sending a *reject* message to the source. Otherwise, the establishment message is forwarded to the destinations.

The following approaches are a selection of current resource allocation schemes:

- In RSVP [17], the resource reservation is done by a *one-pass mechanism.* Once the reservation is established, the reservation request messages are resent periodically to dynamically adapt the change in load. Connections are established without considering QoS requirements of receivers.

- The Berkeley Tenet Suite 2 [3, 12] uses a scheme that is derived from Suite 1 [3], its unicast predecessor. The resource reservation is done by a *two-pass mechanism.* In the first pass, the resources are reserved temporarily, and if the connection is accepted by the network, the amount of allocated resources is adapted to the actual requirements (*relaxed*) in a second pass. As we describe below, this reservation mechanism is not well suited for multicast connections.

- RSVP with OPWA [15] allows nodes along the multicast tree to select levels at which they want to support a multicast connection. Resource reservation is a one-pass process sending from the receivers, and advertisement messages which include the end-to-end delay bound are sent periodically from each source to all receivers. Nodes can evaluate the end-to-end delay resulting from per-link service requests, and choose their desired level of service.

The process of adapting the resource allocation during the second pass of the Tenet Scheme is called *resource relaxation.* Effects of resource relaxation are extended end-to-end deadlines, relaxed jitter requirements, or decreased peak traffic rates or permissible burst sizes. This scheme can be applied to unicast efficiently, but is very cumbersome in multicast, because the amount of resources that can be relaxed in the nodes in one subtree depends on the resource requirements in another subtree, and vice versa. Thus, if relaxation is performed in the second pass, this can lead to low effective resource utilization. We therefore suggest a *three-pass scheme* that effectively relaxes resource requirements along the multicast tree.

This paper is structured as follows. In Section 2, we describe the issues related to providing real-time communication services in a multiparty environment. We describe connection establishment protocols and their interactions with schedulers at the nodes. (In this work, we focus on rate-controlled static priority scheduling at the nodes.) We will describe why traditional resource allocation protocols are not sufficient to support multicasting. In Section 3, we describe a novel three-pass establishment protocol that allows for resource relaxation on multicast connections. We compare the performance with existing establishment protocols in Section 4.

2 Background

In *real-time communication* [1], the quality of communication depends on the time at which messages are successfully delivered to the recipients. Network *Quality of Service (QoS)* requirements are typically formulated in terms of performance metrics such as deadline, delay jitter, and loss rate. The *deadline* is the specific maximum delivery delay bound, which is an application-layer, end-to-end timing constraints. Another important performance metric is *delay jitter*, which is the maximum variation in delay experienced by packets in a single connection. Some real-time applications such as noninteractive television and audio broadcasting require bounds on jitter but not delay. The delay jitter is usually controlled, at cost of additional delay, by using bufferjs.

In recent years, a number of new classes of distributed applications have emerged, such as remote video, multimedia conferencing, data fusion, visualization, and virtual reality. The development from data-oriented to multimedia services, i.e., from non-real-time to real-time service, requires the QoS guarantees instead of best-effort-delivery. A number of real-time communication protocols have been developed that allow the network service to provide the applications with the required QoS guarantees. The basic paradigm used in such protocols is that of the "performance contract" [9]; during a separate connection establishment process, the network service and the application enter an agreement that the network service guarantees the required QoS as long as the traffic generated by the sender does not exceed a specified amount. If necessary, the network service can *police* the traffic stream, that is, drop the excessive traffic.

The mechanisms that enable such a performance contract are the following [9]:

- Connection oriented service: This allows the underlying network to make decisions based on longer-lasting connections instead of individual packets.
- Traffic specification: This provides the underlying service with information about the traffic that is expected to be generated by the source. Since the traffic is very often bursty, appropriate traffic characterization must be possible to allow statistical multiplexing and/or policing at the network service boundary.
- Scheduling at switches and routers: This provides a mechanism to enforce timing requirements by reserving the appropriate amount of resources, where resources can be scheduling priority, bandwidth, or buffer space.
- Admission control: The network must be able to send a busy signal at connection establishment time if the current load does not allow to give the required QoS guarantees to the requesting connection, and, the performance contract could not be maintained.

With these mechanisms in place, the process of determining whether a connection can be admitted to the system is done during a *connection establishment* step, which precedes the transmission of data on the connection.

A new connection can be accepted (and therefore established) only if enough resources are available in the system to guarantee the QoS requirements for the new connection on one side, and existing connections on the other. If so, the necessary resources are *reserved* for the newly established connection.

2.1 Multiparty Communication

Newly emerging applications in distributed multimedia and distributed simulation are increasingly expected to support a multiparty environment, in which connections are set up between a sender and multiple receivers (point-to-multipoint) or multiple senders and multiple receivers (multipoint-to-multipoint) [7]. In order to conserve resources, the underlying network service must be able to support a point-to-multipoint connection abstraction (e.g, hostgroups in multicast IP [6] or target sets in Tenet Scheme 2 [12, 13]) and an appropriate mechanism to support multicasting at network level.

2.2 Real-Time Internetworking Protocols

A number of establishment protocols for internetworks have been proposed in the literature. Some representative approaches are described below.

ST-2 [16] is the successor of ST [11], which in turn was designed to support voice conferencing and is capable of making both unicast and multicast resource reservations. ST reserves resources over a single, duplex distribution tree that is created by merging the paths of multiple unicast routes; this was done with the assumptions that the routes are reversible and the application data traffic travels in both directions. However ST requires a centralized *Access Controller* to coordinate among all the participants and to manage the tree establishment.

The ST-2 protocol continues to create its own multicast trees by blending the paths from unicast routing; however, ST-2 establishes multiple simplex reservations to eliminate the Access Controller. Each data source makes a resource reservation along a multicast tree that is rooted at the source and reaches out to the receivers; the reservation made along the tree uses a single flowspec, therefore ST-2 cannot accommodate heterogeneous receivers. Because each data source makes its reservation independently, a single pipe is reserved from every source to every receiver in the same multicast application group. Thus, neither ST nor ST-2 provide a robust and efficient solution to the multicast resource reservation problem.

In *RSVP* [17], the traffic specification is forwarded from the sender along the multicast tree inform of *path* message. Receivers establish connections by sending *reservation* request in the reverse(i.e. upstream) along the multicast tree. The nodes along the way reserve the necessary resources to accommodate the new request. If the reservation is rejected at any intermediate server, the RSVP *reject* message will be sent back to the receiver, and the reservation message is discarded. Otherwise, the message is propagated upstream. The reservation request messages are sent periodically to adapt to change dynamically.

This scheme can satisfy the per-link services but does not allow end-to-end delay or jitter bounds, because the network cannot tell whether the cumulative delay resulting from the reservation meets the desired end-to-end delay. However, this scheme supports multicast very well by combining IP multicast and a receiver-initiated connection establishment scheme.

In *One-Path-With-Advertisement* (OPWA) [15], the basic RSVP scheme is extended to give receivers and nodes along routes enough information so that they know the effect of local resource allocations on the end-to-end QoS guarantees. This is done by so-called "advertising" (ADV) control message that are sent from the sources to all receivers. ADV message "advertise" the end-to-end QoS guarantees provided with given local resource allocations at the nodes on the route. Nodes on the route evaluate the end-to-end delay resulting from per-link service requests, and choose the desired level of service.

Tenet Suite 2 [4, 13] uses a *two-pass allocation* scheme patterned after Tenet Suite 1 [8, 10], which is extended by providing support for multiparty. Suite 2 uses *target sets* as receiver abstraction, and provides extensive functionality needed for multiparty, such as sender-initiated, and receiver-initiated establishment, dynamic membership in target sets, resource sharing among connections, third-party operations, and advance reservation.

Similarity to Suite 1, the establishment backbone of the protocol is provided by the Real-Time Channel Administration Protocol (RCAP) [13]. The establishment of a real-time multiparty channel happens as follows: First, RCAP gets a possible route for the channel from the routing subsystem. The server then issues a reservation message which contains the traffic specification for the channel. This message follows the given route, replicating itself at each branch node it encounters on its path, and sending a copy of the message down each of the branches of that subtree. Each node on the path reserves resources for the new channel. Given the amount of resources it allocates and the local scheduling policy, it determines arrival of the local delay bound for the channel, and passes the cumulative delay to the next node downstream. Upon arrival of the message at the receiver, the cumulative delay is compared with the end-to-end delay required by the QoS specification of the receiver. If the cumulative delay exceeds the end-to-end delay, the receiver issues a *reject* message, and the connection is denied. Otherwise, a *relax* message is issued and sent back upstream. Upon receiving this *relax* message, each node adjusts the tentative resource allocations made earlier according to the information received from each accepting destination. The process of adjusting the resource requirements and reducing them to satisfy the needs of accepting destinations is called *resource relaxation*. At the end of this process, the source receives the results of the request in a single return message, and transmits them to the client for evaluation and further action. The newly established channel remains in existence until it is torn down by the client.

In case of unicast channels, the two-pass resource allocation procedure is enough to relax the tentative resource allocation. However, such a scheme cannot be used for multicast real-time channels because the interdependence between a branch node and its descendents effectively introduces interdependence relations between resource allocations across the subtrees. To solve these problems, we suggest a *Three-Pass Establishment Protocol*, which will be described in Section 3.

Dynamic Resource Migration (DRM) [5] is a gradient-based dynamic resource reservation protocol using a one-pass establishment protocol. During the establishment, no attempt is made to relax resource allocations; that is, local resource managers allocate resources at will in the attempt to meet the end-to-end requirements. After the connection establishment, resources are dynamically reallocated to balance the load on nodes. In particular, excessive slack detected at the receivers can be "traded" against relaxation of resources at intermediate nodes. This reallocation must be done without affecting the QoS guarantees. The scheme has been proved to effectively relax resource allocations after establishments and to quickly adapt to changes in the network load.

2.3 Packet Scheduling and Admission Control

Admission control and resource allocation are tightly related to the packet scheduling algorithms in the switches and routers. In order to simplify the following discussion, we will assume that the nodes in the network use *rate-controlled static-priority* (RCSP) [18] schedulers. We chose this algorithm because it makes the local admission decision particulary simple to describe. Our scheme works for other scheduling algorithms

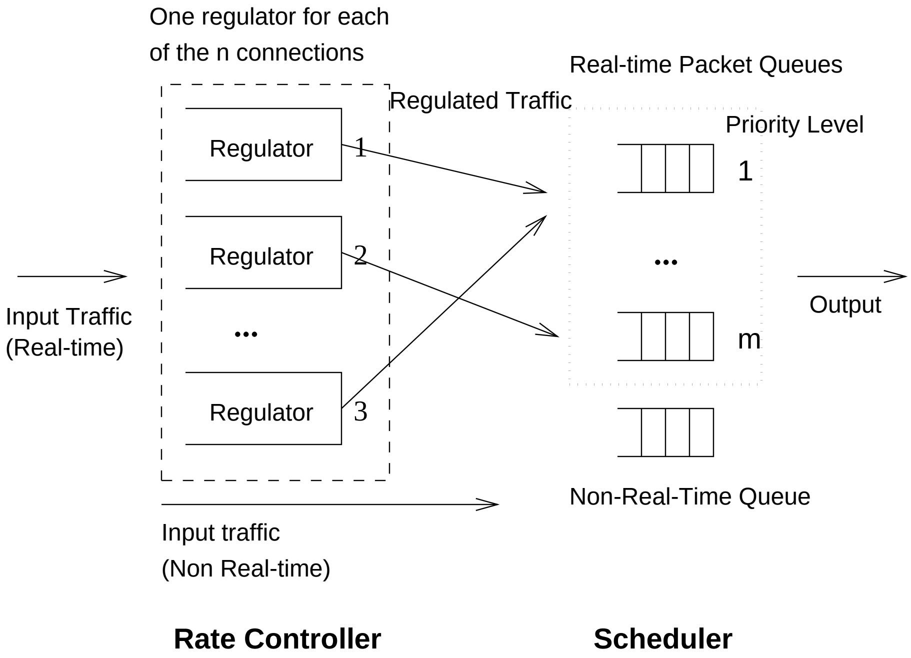


Figure 1: Rate-Controlled Static-Priority Queueing.

as well, such as EDF, or various forms for regulated EDF (jitter EDD or delay EDD), or combinations thereof.

A RCSP server consists of two components, a *rate controller* and *static priority scheduler* as illustrated in Figure 1. The *rate controller* acts as traffic regulator, and shapes the input traffic from each connection into the desired traffic pattern by assigning an eligibility time to each packet. It consists of a set of regulators, one for each of the connections traversing the switch. The regulators control the interactions between switches and control jitter. There are two types of regulators: *rate-jitter controlling regulators*, which control the rate jitter and partially reconstructing the traffic pattern, and *delay-jitter controlling regulators*, which control delay jitter and fully reconstruct the traffic pattern.

The *static priority scheduler* consists of a number of prioritized real-time packet queues. Each connection is assigned to a particular priority level and, thus, to a particular queue, at establishment time. Since schedulers are assigned to control the transmission for particular links, a branch node may have several schedulers over for each outgoing link. Different priorities can therefore be assigned to the same connection at outgoing links of a branch node. During system configuration time, *a priori* local delay bounds are assigned to each priority level. The higher the priority level assigned to a connection, the lower the local delay bound. By limiting the number of connections at each priority level using the admission control conditions in Equation (1) [18], the waiting time of each packet at a priority level is guaranteed to be less than the delay bound associated with that level:

$$\sum_{k=1}^{m}\sum_{j=1}^{i_k}\lceil\frac{d_m}{X_{k,j}}\rceil P_{k,j} + P_{max} \leq d_m l \tag{1}$$

where d_m is the *local delay bound* previously assigned to priority level m of the scheduler. The left side of the inequality defines the maximum amount of time a packet can be delayed by other packets at the same or higher priorities. $X_{k,j}$ and $P_{k,j}$ denote the *minimum interarrival time* and *maximum packet size*, respectively, of the jth connection among i_k connections traversing the switch at priority level k at a particular scheduler. The *link speed* is l, and the size of the largest packet that can be transmitted over the link is P_{max}. If Inequality (1) is satisfied, the waiting time of an eligible packet at level m is bounded by d_m. It follows that Inequality (1) can be used as the admission control condition for each switch.

3 Three-Pass Establishment Protocol

In this chapter, we describe a three-pass resource allocation protocol for multiparty real-time communications. To simplify the following discussion, we assume that all nodes use RCSP schedulers, and that the local admission control mechanisms are designed accordingly. Although the following discussion uses RCSP as a basis, it remains valid for other scheduling algorithms as well.

The three-pass establishment process consists of a request pass, followed by a reply pass, and then by a relaxation pass. During the *request pass*, an *establish request* message is sent along the - previously determined - multicast route from the sender to the receivers of the multiparty connection. Upon receipt of the request message, every node tentatively allocates as many local resources (bandwidth, buffer space, scheduling priority) as possible. The node then forwards the request message down the multicast tree. If the node is a branch node, each descendant is forwarded a copy of the request message.

When the request message reaches a receiver node, the performance requirements specified in the request message are compared in the *final acceptance test* with the ones provided by the connection, and an appropriate reply message is returned to the sender in the second pass, the *reply pass*. What needs to be done when

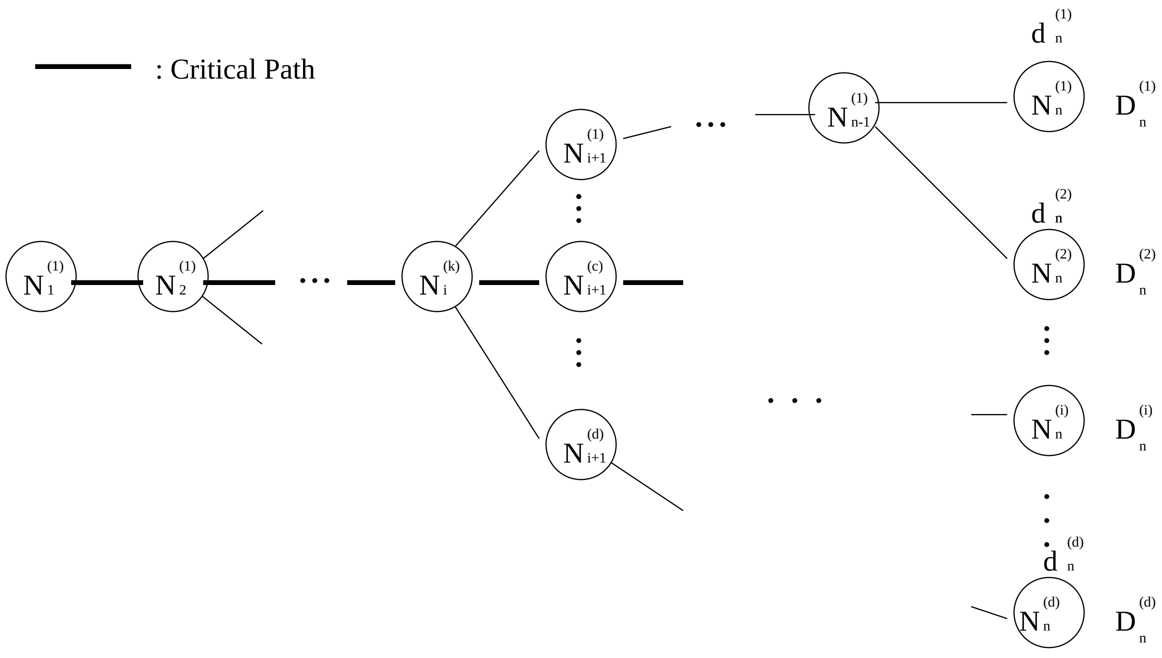


Figure 2: Example of a Multiparty Connection.

a receiver rejects a connection establishment request depends on the establishment semantics (all-or-none establishment, partial establishment, or variations thereof). In the following description, we therefore assume that all receivers anccept the establishment request. In particular, if the performance requirements are met, an *establishment-accept* message is generated and returned; otherwise an *establishment-reject* message. During the reply pass, the establishment reply messages generated at the receiver nodes travel back to the sender. Each node in the multicast tree collects the reply messages from each descendant and forwards a single reply to its parent. The content of this message depends on the result of the establishment request in the subtree and on the semantics underlying the establishment request.

When the reply message reaches the sender, the *relaxation pass* is initiated, and the resources are relaxed from source to destinations. At the end of this pass, each node has allocated the minimum amount of resources necessary to satisfy the performance requirements of the connection. If appropriate priorities are assigned to establishment messages, data transmission can be started immediately after starting the relaxation pass. This reduces the establishment overhead, and has the three-pass mechanism perform as if it consisted only of two passes. In the following, we describe in detail each pass of a channel establishment. To explain this protocol, we will use the example of the multiparty connection in Figure 2.

3.1 The Request Pass

We assume that a client requesting to establish a channel from a source $N_1^{(1)}$ is given a route by the routing subsystem. It then issues an *establish request* message that travels along that route to all d receivers $N_n^{(1)}, \cdots, N_n^{(d)}$. In the first pass, the *traffic parameters* of the connection, such as X_{min}, X_{ave}, I, P, are passed along the given route as part of the *establish request* message, which is replicated at each branch node it encounters on its path, and sent down each of the branches of that subtree.

At each node, admission tests are performed based on the parameters in the establish request message sent from the source. For the case of RCSP, the admission control decision is based on Equation (1). In this first pass, the connection is assigned the highest available priority. In a branch node, the admission tests are performed for each subtree separately. So the allocated priority level may be different for the different outgoing links of a node, according to the resource availability. If all priority queues are full, the connection request is rejected. We note that, in this case, the available bandwidth of the link would be exceeded.

To determine whether enough buffer space is available at the local node, the following equation can be used,

$$B_i = (\lceil \frac{d_{i-1}}{X_{min}} \rceil + \lceil \frac{d_i}{X_{min}} \rceil)P,$$

where d_{i-1} is the local delay bound of the priority level assigned to the connection on the predecessor, and d_i is the local delay bound on the current node. If B_i exceeds the currently available buffer space, the connection is rejected.

If any of the tests fails, a *channel-reject* message is sent back upstream nodes, and the tentative resource reservations for this connection are canceled. If all tests are successful, the *establish request* message is forwarded down the multicast route until it arrives at a receiver. Every node keeps track of the *cumulative delay*, which is the sum of the local delays from the source to the node. If $d_{i-1}^{(k)}$ is the local delay of output link from N_{i-1} to $N_i^{(k)}$, the cumulative delay at Node $N_i^{(k)}$ can be represented as follows:

$$CD_i^{(k)} \quad = \quad \textit{cumulative delay from } N_1^{(1)} \textit{ to } N_i^{(k)}$$

$$= \sum_{h=1}^{i-1} d_h^{(k)}$$

During the request pass, each node on the route is passed the cumulative delay of its predecessor in the route as part of the *establish request* message. This allows the node to calculate its own cumulative delay, which is then forwarded to the descendants.

When the *establish request* message reaches a destination, a final acceptance test is performed. As part of it, the cumulative delay is compared against the end-to-end delay bound D. If $CD_n^{(k)} \leq D_n^{(k)}$ at a destination node $N_n^{(k)}$, the connection is accepted, otherwise rejected, in which case the establishment process is aborted by sending a *establish reject* message sent upstream to the sender.

3.2 The Reply Pass

If the admission test of the destination succeeds, the second pass - the *reply pass* - is initiated. During this pass, an *establish accept* message is forwarded from the destination upstream. As the message travels upstream, each node calculates the *feasible delay* of packets on that node. By the feasible delay $FD_j^{(k)}$ on Node $N_j^{(k)}$, we denote the maximum cumulative delay a packet can experience on nodes upstream without missing the end-to-end delay at the destination. In other words, a packet may miss its deadline at one or more destinations if it is delayed for more than $FD_j^{(k)}$ before arriving at Node $N_j^{(k)}$. The feasible delay on $N_j^{(k)}$ is defined as follows:

$$
\begin{aligned}
FD_n^{(k)} &= D_n^{(k)}, \ \textit{if } N_n^{(k)} \textit{ is a destination} && (2)\\
FD_j^{(k)} &= min(FD_{j+1}^{(1)} - d_j^{(1)}, FD_{j+1}^{(2)} - d_j^{(2)}, \\
&\quad \cdots, FD_{j+1}^{(b)} - d_j^{(b)}), \ \textit{otherwise} && (3)
\end{aligned}
$$

where b denotes the number of successors of $N_j^{(k)}$.

As the *establish accept* packets generated at the destinations travel up the multicast tree, they carry with them the feasible delay of the last node they traversed. Each node N_j waits for the messages from all its successors $N_{j+1}^{(1)}, \cdots, N_{j+1}^{(b)}$ before determining the local feasible delay according to Equation 3, and forwards the message with the new value to the next node upstream. At each node N_j, the successor with the minimum value for its feasible delay determines the direction of the critical path for the subtree with node N_j as root. During the reply pass, every node is informed about the direction of the critical path by comparing the feasible delay values of incoming messages. Each node N_j informs the next node upstream about the length LCP_j of the critical path (in numbers of links) starting at N_j. In summary, upon receiving the establish message from all successor nodes, the message from successor $N_{j+1}^{(k)}$ having the entries $(FD_{j+1}^{(k)}, LCP_{j+1}^{(k)})$, the node N_j performs the following four steps.

1. Determine the successor on the critical path, i.e., the node $N_{j+1}^{(c)}$ with the minimum value $FD_{j+1}^{(c)} - d_j^{(c)}$.
2. Determine the local feasible delay $FD_j = FD_{j+1}^{(c)} - d_j^{(c)}$.
3. Determine the length of the critical path starting at N_j, $LCP_j = LCP_{j+1}^{(c)} + 1$.
4. Forward the establish request message with the updated entries (FD_j, LCP_j) to the next node upstream.

For the example depicted in Figure 2, by equation (2) we have

$$
\begin{aligned}
FD_n^{(1)} &= D_n^{(1)} \\
FD_n^{(2)} &= D_n^{(2)}
\end{aligned}
$$

Let $FD_n^{(1)} - d_{n-1}^{(1)} < FD_n^{(2)} - d_{n-1}^{(2)}$. Then by equation (3), it follows that

$$
\begin{aligned}
FD_{n-1}^{(1)} &= min(FD_n^{(1)} - d_{n-1}^{(1)}, FD_n^{(2)} - d_{n-1}^{(2)}) \\
&= FD_n^{(1)} - d_{n-1}^{(1)}.
\end{aligned}
$$

We note that the link from $N_{n-1}^{(1)}$ to $N_n^{(1)}$ is a critical path, and $N_n^{(1)}$ is set as critical node, $N_n^{(c)}$. Thus,

$$
\begin{aligned}
LCP_n^{(1)} &= 0 \\
LCP_n^{(2)} &= 0
\end{aligned}
$$

$$LCP^{(1)}_{n-1} \quad = \quad LCP^{(1)}_{n} + 1 = 1.$$

We note that the feasible delay FD_1 at the source node indicates how much additional delay can be introduced along the critical path of the connection without missing end-to-end deadlines at the receivers.

3.3 The Relaxation Pass

When the *reply* message reaches the source node, the *relaxation pass* is initiated by sending a *relaxation* message downstream to all destinations. When sent by a node N_j to a successor $N^{(k)}_{j+1}$, this message contains the cumulative delay after relaxation $CD^{'}_{j}$ on all nodes on the path from the source to N_j. During the relaxation pan, each node reduces the priorities assigned to each outgoing link, and thus increases the local delay on each link. The amount by which the priorities can be reduced on a node N_j depends on the reduce for $CD^{'}_{j-1}$ and the feasible delays on the successor nodes.

When a node receives a *relaxation* message, it calculates a *target value*, trd_j, for the local delay after relaxation for each outgoing link. This value is a fraction of the slack $S^{'}_{j}$ available for transmission in the subtree starting at Node N_j after relaxation on upstream nodes. For each successor node $N^{(k)}_{j+1}$, the value for the slack on the corresponding outgoing link is defined as

$$S^{'(k)}_{j} = FD^{(k)}_{j+1} - CD^{'}_{j}.$$

The target value for the local delay on that link is then determined by the next formula:

$$trd^{(k)}_{j} = \frac{S^{'(k)}_{j}}{LCP^{(k)}_{j}}. \tag{4}$$

For each successor node $N^{(k)}_{j+1}$, the lowest priority level on the outgoing link of N_j is determined for which the local delay does not exceed trd_j. Let $d^{'}_{j}$ be the resulting delay. Node N_j first finalizes the priority assignment and then forwards the *relaxation* message to each successor node $N^{(k)}_{j+1}$ with the updated value

$CD_j^{'}$ for the cumulative delay up to N_j after relaxation, where

$$CD_{j+1}^{'(k)} = CD_j^{'} + d_j^{'(k)}.$$

The formula 4 tries to distribute the available slack equally ones the nodes of the critical path of the subtree starting at node N_j. In Chapter 4, we will investigate what the effects of varying this basic allocation scheme are. Specifically, we will generalize the scheme from using an equal distribution to using geometrical distribution, by introducing a *relaxation factor* α. By choosing appropriate α, the relaxed amount of delay is distributed as follows:

$\alpha < 1$: more relaxation at sender side

$\alpha = 1$: identical relaxation

$\alpha > 1$: more relaxation at destination side

$$\begin{aligned}
S_j^{'(k)} &= trd_j^{(k)} + trd_{j+1}^{(c)} + \cdots + trd_{j+LCP_j^{(k)}}^{(c)} \\
&= trd_j^{(k)} + \alpha trd_j^{(k)} + \cdots + \alpha^{LCP_j^{(k)}-1} trd_j^{(k)} \\
&= trd_j^{(k)} \sum_{h=0}^{LCP_j^{(k)}-1} \alpha^h \\
&= trd_j^{(k)} \frac{\alpha^{LCP_j^{(k)}} - 1}{\alpha - 1}
\end{aligned}$$

The formula in 4, therefore becomes

$$trd_j^{(k)} = \frac{S_j^{'(k)}}{\sum_{h=0}^{LCP_j-1} \alpha^h}.$$

We note that for $\alpha = 1$, the allocations are identical to the one defined in formula 4.

4 Simulation and Experiments

We assessed the quality of the three-pass protocol described in this work we performed a suite of simulation experiments. The goal of these experiments was to determine the effect of this protocol on the *Call-Blocking Probability*, that is, the probability of a request for connection establishment being rejected.

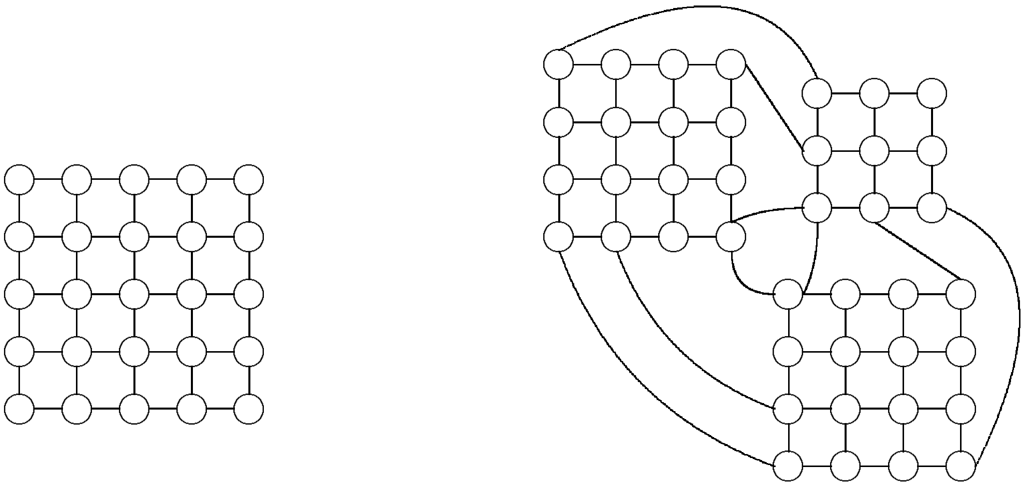
In the experiment suite, we simulated the establishment and teardown of one-to-many uncompressed CD-audio connections with a varying numbers of destinations. Specifically, the traffic specification for the connections is that of constant-rate streams with maximum packet size of 1KB and minimum interarrival time of 13ms, for a local bandwidth requirement of 0.7Mb/s. The end-to-end delay requirement is 188ms.

The underlying network consists of nodes connected by OC-12 level links (620Mb/s). The nodes use RCSP schedulers to control the traffic on the outgoing links. The schedulers have 6 priority queues, which are configured so as to have delay bounds 5, 14, 27, 40, 66, and 100 packet transmission times, respectively. According to the acceptance test criteria described in Section 2.3, each outgoing link can carry up to 4 connections at priority 1, up to 6 connections at priority 2 and above, 8 connections at priority 3 and above, 9 connections at priority 4 and above, 10 connections at priority 5 and above, and 12 connections at priority 6 and above. Each node has a 30MB buffer to store the packets temporarily in the traffic regulator and the priority queues.

Requests for connection establishment form a Poisson process with rate λ, and life times for connections identically distributed with the common distribution being exponential with mean $1/\mu$. In the following we will keep $1/\mu$ fixed to one normalized time unit, and will let λ vary.

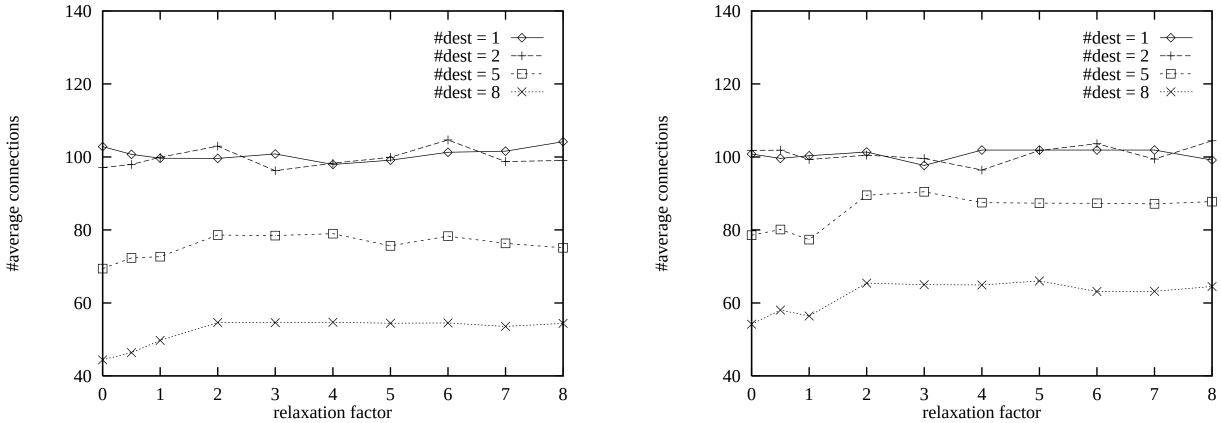
An establishment request for a connection can be rejected for three reasons; first, the end-to-end delay violates the guarantees at a destination, second, a node on the route cannot allocate enough buffer space, and third, the bandwidth on a link is exceeded. If one of the destinations or a node on its path rejects the connection, all temporarily reserved are freed.

We note at this point that if a connection is rejected for lack of bandwidth, this is not caused by inadequate allocation of resources, but is issues of inappropriate routing. Since we are no addressing routing in this work, in the following experiments we ignore establishment requests that are rejected because of lack of available bandwidth. The performance metric in these experiments is the average number of accepted connections in the system at any time in steady state. The experiments were performed for two topologies, depicted in Figure 3. Individual connections were guaranteed by randomly locating senders and receivers, and connecting them with shortest routes from the senders to the receivers.



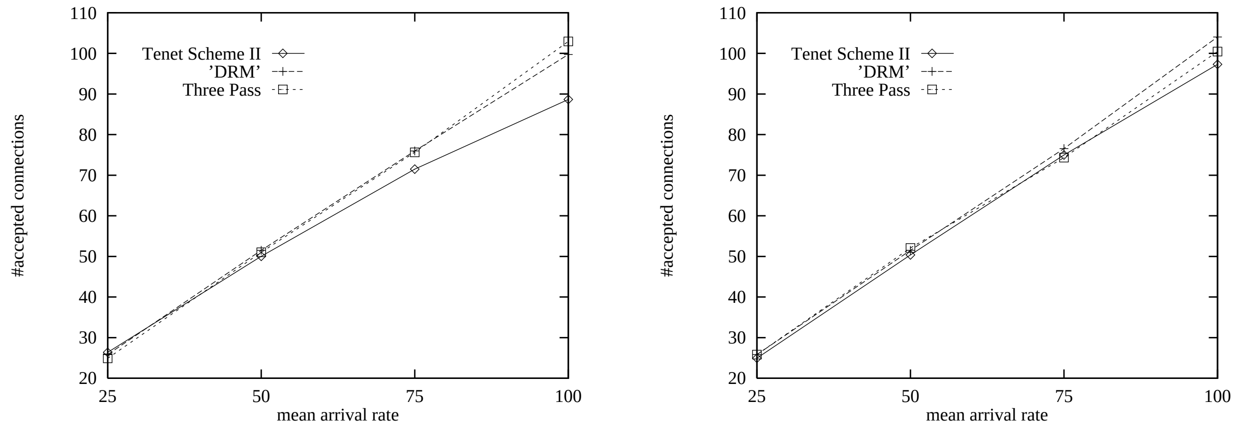
(a) Topology 1 (b) Topology 2

Figure 3: Network Being Simulated.



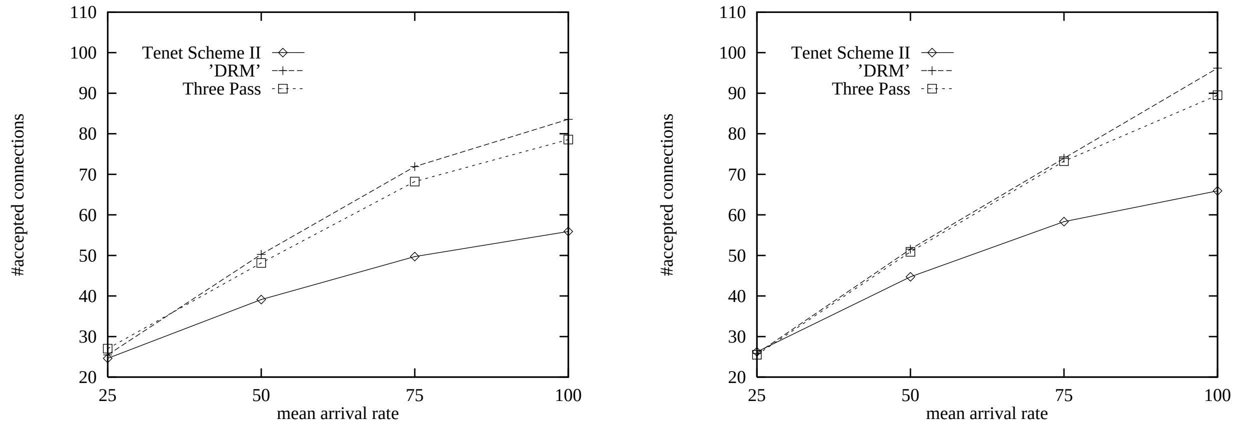
(a) Topology 1 (b) Topology 2

Figure 4: Average Number of Connections in Steady State.



(a) Topology 1 (b) Topology 2

Figure 5: Average Number of Accepted Connections (Two Destinations).



(a) Topology 1 (b) Topology 2

Figure 6: Average Number of Accepted Connections (Five Destinations).

4.1 Effects of Relaxation Factors

In a first series of experiments, we studied the effect of the relaxation factors α, i.e., the factors determining whether resources should be relaxed more at the source or at the receivers, on the performance of the algorithm. In these experiments, we keep the arrival rate λ constant to be 100 times the service rate. Figure 4 shows the average numbers of accepted connections for a varying value if α and increasing numbers of destinations per connection. In the case of one of two destinations per connection, all connections are accepted. As to be expected, the acceptance rate decreases with increasing numbers of destinations per connection, as the connections put more load on the system.

For connections with one or two destinations, the value of α has virtually no effect on the call blocking probability. This is because uneven resource allocation cancel each other out in average for these essentially unicast connections.

As the number of destinations per connection increases, the effect of varying α becomes more pronounced. The results shows that the acceptance rate is higher for a relaxation factor larger than 1. In this case, more resources are relaxed at the destinations, since there are many more destinations than senders, the total amount of resources used per connections is reduced with increasing value for the relaxation factor α.

We notice that very large (larger than 2.0) or very small (smaller than 0.5) values for α do not significantly influence the acceptance rate. This can be explained because the relaxation factor determining the calculation of the target delay *trd* at the node, which gives only an *indication* on how much resources can be relaxed. The actual overall amount of resources freed depends on the current resource use at that node, and the remaining portion is carried forward for relaxation on the next node downstream. According to the results on these experiments, we identified a value for α of 2.0 to give good resluts, and will therefore use it in the following experiments.

4.2 Comparison with other Allocation Schemes

In order to assess the quality of our resource allocation scheme, we compared it with two other, comparable protocols. Figures 6 and 5 depict connection acceptance rates as functions of increasing connection arrival rates for varying numbers of destinations per connection. In each figure, we compare the connection acceptance rate of the Three-Pass protocol proposed in this work with the traditional Tenet Suite 2 [13] establishment protocol, and the Dynamic Resource Migration protocol [5] (DRM).

As described earlier, the traditional Tenet Scheme 2 protocol is a two-pass establishment that does not perform resource relaxation. DRM is a one-pass establishment protocol that relies on a dynamic local balancing scheme to adapt the resource allocations dynamically to the changing load in the network. As to be expected, the three-pass protocol performs better than the basic Suite 2 protocol, in particular for situations with high load, either because of high arrival rates or a large number of destinations. This is because Suite 2 does not perform relaxation, which quickly saturates high priority queues, and therefore wastes resources. On the other side, DRM consistently performs slightly better than the three-pass protocol.

When a new connection is established, DRM uses load balancing to relax the resource allocations for that connection. In addition, DRM adapts the resource allocations dynamically to reflect the changing load in the system as new connections are established or existing connections are torn down. We believe that the increase in performance of DRM over the three-pass algorithm is due mostly to the dynamic behaviors of DRM. However, the data collected in Figures 6 and 5 indicates that the small increase in performance may not warrant the additional complexity of running a dynamic resource reallocation scheme.

References

[1] C. Aras, J. Kurose, D. Reeves, and H. Schulzrinne, "Real-time communications in packet-switched networks," in *Proceedings of the IEEE*, vol. 82, No. 1, pp. 122-139, Jan. 1994.

[2] A. Ballardie, P. Tsuchiya, and J. Crowcroft, "Core based trees (CBT)", IETF Draft, Nov. 1992.

[3] A. Banerjea, D. Ferrari, B. Mah, M. Morgan, D. Verma, and H. Zhang, "The Tenet realtime protocol suite: design, implementation, and experience," Tech. Rep. TR-94-059, International Computer Science Institute, Berkeley, California, Nov. 1994.

[4] R. Bettati, D. Ferrari, A. Gupta, W. Heffner, W. Howe, M. Moran, Q. Nguyen, and R. Yavatkar, "Connection establishment for multiparty real-time communication," in *Proceedings of the 5th International Workshop on Network and Operating System Support for Digital Audio and Video*, Durham, New Hampshire, Apr. 1995.

[5] R. Bettati and A. Gupta, "Dynamic resource migration for multiparty real-time communication," in *Proceedings of the 16th International Conference on Distributed Computing Systems*, Hong Kong, May 1996

[6] S. Deering, "Host extentions for IP multicasting," Request for Comments RFC-1112, Aug. 1989.

[7] S. Deering, "Multicast routing in a datagram internetwork," Tech. Rep. No. STAN-CS-92-1415, Stanford University, California, Dec. 1991.

[8] D. Ferrari, "Real-time communication in packet switching wide-area networks," Tech. Rep. TR-89-022, International Computer Science Institute, Berkeley, California, May 1989.

[9] D. Ferrari, "Client requirement for real time communication services," *IEEE Communications Magazine*, vol. 28, no. 11, pp. 65-72, Nov. 1990.

[10] D. Ferrari, A. Banerjea and H. Zhang, "Network support for multimedia: a discussion of the Tenet approach," in *Computer Network and ISDN Systems*, pp. 1267-1280, July 1994.

[11] J. Forgie, "A proposed Internet stream protocol," Internet Experimental Notes IEN-119, Sep. 1979.

[12] A. Gupta, W. Howe, M. Moran, and Q. Nguyen, "Resource sharing in multi-party realtime communication," in *Proceedings of IEEE Infocom '95*.

[13] B. Mah, "A mechanism for administration of real-time channels," Tech. Rep. UCB/CSD-93-735, University of California, Berkeley, California, Mar. 1993.

[14] C. Patridge, "A proposed flow specification," Internet RFC-1363, July 1992.

[15] S. Shenker and L. Breslau, "Two aspects of reservation establishment," in *Proceedings of SIGCOMM'95*, Cambridge, Massachusetts, Aug. 1995.

[16] C. Topolcic, "Experimental Internet stream protocol: version 2 (St-II)," Internet RFC-1190, Oct. 1990.

[17] L. Zhang, S. Deering, D. Estrin, S. Shenker and D. Zappala, "RSVP: a new resource reservation protocol," *IEEE Networks Magazine*, vol. 31, No. 9, pp. 8-18, September 1993.

[18] H. Zhang and D. Ferrari, "Rate-controlled static-priority queueing," in *Proceedings of IEEE INFOCOM'93*.