Resilient Multicast Support for Continuous-Media Applications

X. Rex Xu Andrew C. Myers Hui Zhang
Carnegie Mellon University
{*rexx, acm, hzhang*}*@cs.cmu.edu*

Raj Yavatkar
Intel Corporation
yavatkar@ideal.jf.intel.com

Abstract

The IP multicast delivery mechanism provides a popular basis for delivery of continuous media to many participants in a conferencing application. However, the best-effort nature of multicast delivery results in poor playback quality in the presence of network congestion and packet loss. Contrary to widespread belief that the real-time nature of continuous media applications precludes the possibility of recovery of lost packets using retransmissions, we have found that these applications offer an interesting tradeoff between the desired playback quality and the desired degree of interactivity.

In particular, we propose a new model of multicast delivery called *resilient multicast* in which each receiver in a multicast group can decide its own tradeoff between reliability and real-time requirements. To be effective, error recovery mechanisms in such a model need to be both fast (due to the real-time constraint) and have a low overhead (due to high volume of continuous media data).

We have designed a resilient multicast protocol called STORM (STructure-Oriented Resilient Multicast) in which senders and receivers collaborate to recover from lost packets using two key ideas. First, group participants self-organize themselves into a distribution structure and use the structure to recover lost packets from adjacent nodes. Second, the distribution structure is dynamic and a lightweight algorithm is used to adapt the structure to changing network traffic conditions and group membership. We have implemented STORM in both VAT and a packet level simulator. Experimental results using both the MBONE and a simulation model demonstrate the effectiveness of our approach.

1 Introduction

Availability of IP multicast has prompted the development of a suite of continuous media applications (nv, vat, vic, ivs) to support multimedia conferencing over the Internet. Because delivery of packets in real-time is essential for continuous playback of audio/video streams, it has been widely believed that it is not important or even feasible to attempt to recover from lost packets by retransmitting them. Instead, it has been observed that continuous media applications can tolerate a certain amount of packet loss and, therefore, most of the research in audio/video transport has concentrated on devising adaptation techniques that minimize the effect of packet loss and variable delays on the quality of audio/video playback. However, we make two important observations about the requirements of such applications.

First, there is a tradeoff between the desired level of playback quality and interactivity. The amount of interactivity within an application has an important bearing on the flexibility in packet delivery times. When an application requires a high degree of interactivity, real-time delivery of packets is essential. However, the desired degree of interactivity varies significantly across different applications. At one extreme, a recording application involves no interactivity. Today, many MBONE tools are used mostly for broadcast events, where there is little interaction among participants and most receivers are only passive listeners. In such cases, playback quality can be greatly improved by waiting for delayed packets and by recovering lost packets using retransmissions. On the other hand, if a high degree of interactivity is desired, the packet delivery delay must be very small, which means that lost packets cannot be retransmitted and packets with large delays must be discarded.

Second, receiver requirements vary widely in terms of tolerable playback quality and desired degree of interactivity. Within the same application, the desired degree of interactivity typically varies from one participant to another. For instance, even in a conferencing session such as an IETF meeting using MBONE tools, there are usually many more passive participants than active participants. Passive participants would rather prefer high playback quality, which can be achieved by a combination of delayed playback and retransmission of lost packets.

Existing continuous media MBONE applications, designed only to trade quality for interactivity uniformly over all participants, fail to take this tradeoff into account and never allow retransmission. We propose a new model of multicast delivery called *resilient multicast* that allows each receiver in a multicast group to independently decide its own tradeoff between desired quality (reliability in delivery) and interactivity (latency in delivery).

In this paper, we study error recovery strategies for resilient multicast. Since the amount of data transmitted by continuous media applications is large, it is desirable to reduce the relative overhead of error recovery packets. In addition, since resilient multicast has real-time requirements, it is important to minimize the delay in recovering lost packets. We propose the Structure-Oriented Resilient Multicast (STORM) protocol that is based on two key ideas: First, group participants self-organize themselves into a distribution structure and use the structure to recover lost packets from adjacent nodes. Second, the distribution structure itself is built dynamically using a lightweight algorithm that takes into account changes in network traffic conditions and multicast group membership.

We have evaluated and verified the effectiveness of our approach using both the experiments across MBONE and a simulation model. The rest of this paper is organized as follows. We first begin with a description of the resilient multicast and how it differs from the conventional notion of reliable multicast (Section 2). Section 3 summarizes Scalable Reliable Multicast (SRM) [9], a well-known protocol for achieving large-scale, reliable multicast using receiver-driven error recovery, and points out its relevance to our work. We then describe the details of a new protocol called STORM (Structure-Oriented Resilient Multicast) based on our approach (Section 4). Section 5 describes the results of a systematic performance evaluation using both MBONE experiments and large-scale simulations.

2 Characteristics of Resilient Multicast

To motivate the need for resilient multicast, we will first discuss the important differences between resilient and reliable multicast. Reliable multicast transport is typically needed by a data dissemination application such as a whiteboard (e.g. *wb*) where a sender wishes to reliably deliver all the parts of its presentation to all the receivers. Many researchers [9, 13, 26, 25, 10] have examined ways of reliably delivering data to all the recipients within a multicast group. Resilient multicast, on the other hand, is directed at continuous media (CM) applications, creating significant differences in the delivery requirements of the two sets of applications:

1. In the wb case, every member requires a uniform level of reliability (all the packets must be delivered eventually) whereas in the case of CM applications, there are real-time requirements, and each member has its own reliability requirement based on its playback delay.

2. In wb applications, traffic tends to be more bursty than in CM applications and overall data rate tends to be much lower than in CM applications. This has important consequences. First, enough bandwidth can be assumed to be available for retransmissions within a wb session. Traffic is continuous with CM applications, bandwidth requirements are high, and the bandwidth consumed by retransmissions can interfere with normal transmissions. Second, to avoid contributing to traffic congestion and interference, resilient multicast must also rely on localized recovery (recovery using retransmissions from nearby group members) as much as possible.

3. In the wb case, each participant maintains persistent state of the session (e.g., buffers all the slides in a presentation) as the sender may refer back and forth to different parts of the presentation. Thus, every member is equally capable of helping other members with recovery from data they have not received. This is not the case with CM applications. Since every member chooses its own playback delay (and, therefore, the amount of data it buffers), a participant's ability to help with error recovery varies widely.

To motivate our design further, we describe the previously used techniques for reliable multicast in the next section and point out the need for a new approach for achieving resilient multicast.

3 Reliable Multicast Protocols

Loss detection and retransmission strategy are two important aspects in the design of any reliable protocol. In a traditional point-to-point reliable protocol such as TCP, positive acknowledgements are used to detect loss and the sender is responsible for retransmission of the packet. In a multipoint communication, if multiple receivers send back positive acknowledgments, the sender would become a bottleneck. This problem is known as an ACK implosion. Therefore, reliable multicast protocols rely on a combination of negative acknowledgements (NACKs) and selective retransmissions to recover from lost packets. In addition, some form of NACK aggregation may be used to further increase efficiency. Among several reliable multicast protocols proposed in the literature, Scalable Reliable Multicast (SRM) protocol [9], used in the popular Internet *wb* program, is a good example of such a protocol. We will use SRM's error recovery mechanisms to motivate our design of a protocol for resilient multicast. In the following subsection, we provide an overview of the recovery scheme used in the SRM protocol.

3.1 Overview of the SRM Protocol

In SRM, whenever a receiver detects a packet loss, it multicasts a NACK packet to the entire group. Upon receiving the NACK packet, any member in the group having the packet can multicast a repair packet. To avoid duplicate NACK and repair packets, a suppression algorithm is used in which a node sets a random timer before multicasting a NACK or repair packet. If a node receives the same NACK or repair packet from another node before the timer expires, it will cancel the planned multicast. Such an algorithm is simple and robust. However, the algorithm will end up consuming a lot of bandwidth when there is little correlation of losses among receivers. For example, in a group of 1000 receivers, even when only one receiver loses a packet, all 1000 receivers need to process the multicast NACK and repair packets. This introduces significant overhead for both the network and the receivers. The problem becomes more severe as the number of receivers in the group increases because the probability that at least one receiver does not receive a multicast packet also increases. In the worst case, for every multicast packet, at least one receiver does not receive the packet, which means *every* packet needs to be transmitted to the whole group at

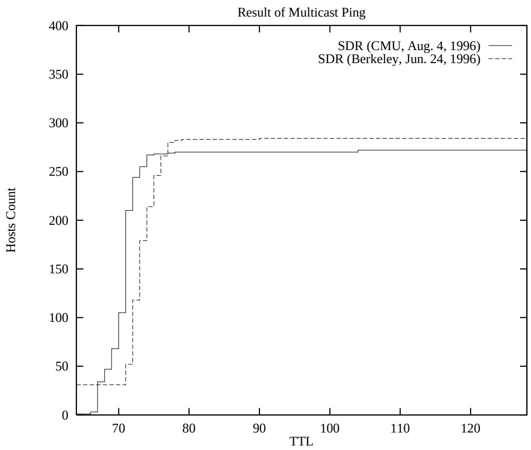


Figure 1: Number of hosts reachable by a given TTL

least twice – this is assuming that the NACK and repair suppression algorithm work perfectly. In reality, in order to reduce the number of duplicate packets, the random timer needs to be set to a rather large value, delaying repair and NACK packets further. This is undesirable for CM applications where packets will be useless after a certain fixed delay.

3.2 Improvements in the SRM Approach

One possible method of improving SRM's efficiency is to use localized recovery. The idea is to multicast NACKs and repairs locally to a limited area instead of to the whole group. Clever use of local recovery can improve the protocol performance in terms of both latency and bandwidth consumption, but requires an effective multicast scope control mechanism.

Using the TTL (Time To Live) field in the IP packet header is one possible way to implement scope control. Each link at a multicast router is configured with some threshold value so that only those packets with TTL values greater than or equal to the threshold can be forwarded to the next hop. As in a traditional IP router, TTL is also decreased by one at each multicast router.

To evaluate how well TTL can be used to control multicast scope, we performed experiments with the *mping* program. We *ping*'ed the popular *sdr* address (224.2.127.254) with different TTL values, and then counted the hosts reached with these TTL values. Figure 1 shows the results for two experiments performed at two hosts, Berkeley and CMU. The figure only shows TTL values from 64 to 128 because when the TTL is less than 64, only hosts within a local region can be reached, and hosts in the same region usually share the same packet loss pattern, reducing the probability that they will be able to provide repairs to each other.

From the figure, it can be seen that TTL is not a good measure of locality. The number of reachable hosts does not increase linearly with TTL values. Rather, the curve has large jumps for certain TTL values and remains flat for other TTL values. In addition, our experiments showed that the TTL values between two hosts are usually not symmetric. That host A can reach host B with some TTL value does not imply that host B can reach host A with the same TTL. When a group member receives a NACK message with a specific TTL value, it does not know which TTL value to use to multicast the repair. To make matters worse, there might be some other members who have also received the same NACK and suppressed their own NACKs. The repairer is expected to provide repair to these members also, but it has no way of knowing the appropriate TTL value to use to reach them all.

To summarize, the SRM protocol represents a simple and robust approach for large-scale recovery based on persistent state, suppression of duplicate NACKs and repairs, and global retransmissions. However, global recovery may incur large overhead both in terms of network bandwidth and end system processing time. In addition, effective suppression in a large group requires higher timer values, which will increase recovery latency. Localized recovery using TTL to limit the scope of multicast NACKs and repairs does not work well. Other approaches for limiting the scope of multicast recovery such as modifications to the underlying multicast routing algorithms need further investigation. Next, we will describe STORM, which is designed to overcome these limitations and provide quick recovery from lost packets.

4 Structure-Oriented Resilient Multicast

In designing error recovery algorithms for resilient multicast, we want to (a) minimize the overhead of control packets as continuous media applications already consume relatively large amount of bandwidth, and (b) minimize the delay in recovery as packets arriving late will be discarded. In the previous section, we argued that multicasting every NACK and repair packet to the entire group will result in large overhead. This will be verified by experiments presented in Section 5. While local multicast may alleviate this problem, the TTL-based scoping mechanism has a number of drawbacks that severely limit its use.

In this section, we present our solution, STORM.

4.1 Overview of the Protocol

A key idea in STORM is to distribute the NACK and repair packets along a structure that is overlaid on application endpoints. Unlike other distribution structures such as the multicast routing tree where interior nodes are routers, both interior and leaf nodes are application endpoints in our structure. The construction and maintenance of the structure is lightweight in the sense that the algorithm keeps a small amount of local state and avoids the use of distributed and dynamic state that needs to be consistent. As will be explained later in this section, besides the packets in the playback buffer, each node needs to keep the following STORM related state: a short list of parent nodes, a quality estimator for each parent node, a level number, a delay histogram of all packets received by the node, a list of timers for the NACK packets that are sent to its parent, and a list of NACK packets from its children for which repair packets have not been sent. All state information is local except the list of NACK packets, which is shared by a pair of child and parent nodes. For the NACK-related state, the only possible inconsistency occurs when a NACK or repair packet is lost, causing a parent and child disagree about which packets the child lacks, in which case the child will timeout and initiate another request. As a result, most decisions are local and only a small number of control packets are needed to maintain the structure. By not guaranteeing full reliability, we are able to avoid some expensive operations such as keeping track of exact group membership and aggregating positive acknowledgments to verify delivery to all group members.

The recovery algorithm works as follows. Each receiver maintains a list of parent nodes, which are other members in the multicast group. Whenever a receiver discovers a packet is missing, it selects one node from the parent list and sends a NACK for that packet to the parent. If, after some timeout, the host has not received a repair, it will choose another parent on the list to which it will send another NACK. This goes on until the packet is recovered or the packet becomes obsolete and there's no need to recover it any more. The purpose of using multiple parents instead of a single parent is to balance load among different members and to make the recovery mechanism more robust.

When a parent receives a NACK from a child for a packet that the parent has in its buffer, it will unicast the packet to the child immediately. If the parent does not have the packet, it will wait for the packet to arrive, and then send it to the child. The packet may not have arrived at the parent for two reasons: either because the packet from the source is delayed, or because the packet was lost and the parent itself is in the process of recovering the packet from its own parent.

In our protocol, NACK and repair messages are sent using unicast UDP except in a LAN environment, where the cost of a multicast and unicast is identical. We avoid large scale multicast of NACKs and repairs to reduce recovery overhead. In addition, NACK and repair packets are sent immediately so that recovery latency is minimized, which is critical for continuous media applications using resilient multicast service.

In the following sections, we describe four key aspects of STORM:

1. a lightweight and scalable protocol to build and maintain the structure distributedly;

2. an algorithm for choosing parents based on the semantics of resilient multicast;

3. a distributed loop avoidance mechanism;

4. a technique for adapting the structure to changes in network conditions and multicast group membership.

4.2 Building the Recovery Structure

The recovery structure is a multi-parent tree with application endpoints being both the interior and leaf nodes. The structure is incrementally built as receivers join the multicast group.

When a receiver first joins the group, it uses an expanding ring search (ERS) technique to look for potential parents. An ERS consists of sending out queries to the multicast group with increasing TTL values.

Members that are already part of the structure can answer ERS queries they receive with a unicast reply. The reply packet contains an indication of the perceived loss rate as a function of the playback delay. The next section describes how a new receiver uses this information to select a parent. When the new receiver collects enough candidates for parent, it stops its ERS.

Note that the ERS query is the only control message that uses multicast in our protocol and TTL values are used to limit the multicast scope, ensuring that we locate suitable parents in a scalable manner. Another advantage of ERS is that usually, adjacent nodes are found as parents, creating a reasonable structure. However, to make the structure optimal for error recovery, parent nodes should be selected carefully. The selection procedure is described below.

4.3 Selection of Parent Nodes

Each receiver keeps track of a set of measured packet losses as a function of the playout buffer size. The loss rate is a monotonically decreasing function of the playback buffer size as a larger playback buffer size will allow packets experiencing longer delays and retransmitted packets to arrive in time to be played back. When a new member is looking for a parent, it knows how long a packet arrival can be delayed and still be useful (based on its own playback buffer size) and therefore, can use a candidate parent's loss rate to determine whether to accept it as a parent or not. As an extremely oversimplified example, consider a receiver with a 200 ms buffer that is looking at two potential parents, A and B. Suppose A received 90% of its packets within 10 ms and 92% of its packets within 100 ms, while B received 80% of its packets within 10 ms and 95% of its packets within 150 ms, the receiver should choose B as its parent. It does not matter how slowly a packet arrives as long as the packet arrives in time to be useful.

The data source timestamps each packet with the time the packet was sent. When a member receives the source's packet, it finds the *adjusted arrival time*, t_a, which is the difference between the time the member received the packet and the timestamp inside the packet. The member keeps a histogram recording how many packets it received for each value of t_a. When the loss rate at time t, $L(t)$, is requested, a member returns

$$1 - \frac{\sum_{i=0}^{t} number\ of\ packets\ for\ which\ t_a = i}{total\ number\ of\ packets\ expected}.$$

To choose from a group of potential parents, the child uses a potential parent's value of $L(t_a + B - d_{parent,child})$ where B is the child's buffer size (in milliseconds), $d_{parent,child}$ is the one–way propagation delay from the parent to the child, and t_a is the typical adjusted arrival time experienced by the child. Note that t_a is computed relative to the child's time frame, so we need to convert it to the parent's time frame.

To make the conversion, we need to find the clock offset between the potential parent and the child. When the child sends the parent a message, it timestamps the message (call this value t_1). The parent records the time it receives the message (t_2) and includes this value in its reply to the child, which it will also timestamp (t_3). The child records the time at which it receives the parent's reply (t_4). Now the clock offset can be computed as half of the sum of offsets between (t_1, t_2) and (t_4, t_3), as:

$$offset = \frac{t_1 - t_2 + t_4 - t_3}{2}$$

Finally, the child can ask its parent to return $L(t_a - offset + B - d_{parent,child})$.

The loss statistic used to judge a parent's suitability is a unique aspect of STORM designed to mesh well with our goal of resilience.

4.4 Loop Avoidance

Since the recovery structure is built in a distributed manner, loops can be formed during the ERS. Loops are undesirable because they can prevent recovery from packet loss. For example, when a loop is formed among a few receivers, any packet loss shared by these receivers will not be recovered. Therefore, STORM should build a structure that is a loop-free multi-parent tree rooted at the data source.

To retain the distributed nature of STORM, our goal was to design a loop avoidance scheme that does not require active coordination and exchange of state information among nodes in the tree. Each node is assigned a level number to reflect its position in the tree. A new node can only choose nodes with a level number lower than its own as its parents. The root has the lowest level, which is 0. The level should be assigned to the node soon after it joins the multicast group, and should not change during its lifetime. It is also desirable that the level should somehow reflect the network topology; for example, those closer to the root should have a lower level than those further away.

We have considered two methods to assign levels to hosts:

- Use hop count to the root as level. When a node joins the multicast group, it tries to find out its hop count to the root and use it as its level. Hop count is usually static (assuming the network routing does not change very frequently) and to some extent reflects the distance between two hosts. However, the current Unix socket API makes it difficult for a user level application to get access to the TTL information, which encodes the hop count, in an IP datagram header.

- Use the estimated round-trip time (RTT) to the root as level. In our implementation, we use the estimated RTT from a node to the root as a substitute for the hop count. Since RTT can be estimated with one packet exchange, each node can send a message to the root when it joins the group to determine its level. (To solve the implosion of such requests when many new members join the group at the same time, each member can wait for a random time before it sends the message to the root.) A minor flaw in using the RTT estimate is that it depends on the network load and may not represent accurately the distance between two hosts. This is acceptable most of the time, since it is not necessary and not possible to make the level always an accurate estimation of the network topology.

Sometimes more than one node may have the same level. In most cases, this is acceptable, but when a large number of hosts have the same level, the number of eligible parents for these hosts can be unacceptably small. To avoid such a situation, each receiver's level is augmented with a random number, known as the minor level. When two hosts have the same level, the minor level is used as a tie breaker with lower minor level deciding which node gets to be the parent.

Our experiments show that the level-based mechanism prevents loops successfully and helps in building a structure that reflects the physical network topology. Figure 3 shows a typical structure built in one of our experiments, with level numbers marked beside each node.

4.5 Adapting the Structure

The structure, once built, is not final. As network conditions and packet loss characteristics change over time, a parent may fail to provide timely recovery in many cases. One possible cause of degraded parental performance is an increase in the loss rate in some part of the network. In the extreme case, a parent simply leaves the group so that it no longer provides any repairs at all.

To adapt to changing conditions, each receiver periodically reevaluates the quality of each of its parents by computing the ratio of the number of repairs supplied by a parent to the number of NACKs sent to the parent. This ratio, which serves as a dynamic rating of a parent's performance, is updated by periodic exponential smoothing. The ratio is checked periodically and if it drops below some threshold, then the parent is removed from the parent list. To find a replacement, a node can start another round of ERS to find a better parent or select a node that was found in a previous ERS but not yet used as a parent.

The ratio of NACKs sent to repairs received combined with the metrics used to choose the parents is used to rank parents in the parent list. The number of NACKs sent to a parent is proportional to its rank in the parent list. For example, if two parents have ranks 1 and 2, then the latter will receive twice as many NACKs as the former. Dynamic adaptation is an important feature of STORM; successful adaptation makes the protocol more robust in the presence of changing group membership and network conditions.

4.6 Distributed State Management

The main difficulty in making distributed algorithms scalable and robust is maintaining distributed state that is both dynamic and consistent. In our design, we try to minimize the amount of state information kept at each node. In addition, we avoid dynamic and distributed state that needs to be consistent across several nodes. For example, in our protocol, while a child node keeps track of its parent nodes, a parent node does not keep track of its children. The parent–child state information is kept at the child instead of at both the parent and child. Another example is the design of the loop avoidance algorithm where each receiver acquires a level number that remains unchanged during the receiver's lifetime. In the protocol, node A can pick node B as its parent only if node A has a larger level number than node B. Since the level numbers define a partial ordering relationship that is time invariant, change of a parent node becomes a local operation.

In contrast, with Lorax, parent nodes explicitly keep a list of children, and children also keep a list of their siblings. To ensure loop-free structure, each node includes its identifier in messages where the identifier encodes the path from the root to the node. Since the path changes dynamically, when a node changes parent, all descendants have to be notified and the information has to be updated after each change in membership. Compared to the method in Lorax, our algorithm is much simpler.

5 Evaluation

To evaluate the performance of STORM, we implemented both STORM and SRM in the audio program *vat*, and conducted experiments over the MBONE. In addition, we implemented both protocols in a packet level simulator, which allows us to compare STORM and SRM in a controlled environment and to evaluate STORM's scalability.

5.1 Performance Metrics

We evaluate the effectiveness of an error recovery protocol along two dimensions: the performance improvement to the application, and the overhead incurred by the protocol. In the context of resilient multicast, application performance is characterized by the loss rate given a fixed size playback buffer. We use two performance indices: the *initial loss rate* of a receiver is the fraction of packets that are originally multicasted by the sender but not received in time by the receiver, the *final loss rate* of a receiver is one minus the fraction of packets that arrive in time for playback among all the packets that are originally multicasted by the sender. Multiple packets with the same sequence number are counted just once. The application performance is characterized by the final loss rate. The lower the final loss rate, the better the performance. Since the final loss rate also depends on the network conditions during experiments, we use the initial loss rate as a reference.

There are two types of overhead, the network bandwidth consumed due to the transmission of NACK and repair packets, and the time spent to process these packets at each node. The processing overhead at a node can be measured by the number of NACK/repair packets sent and received by the node. The bandwidth

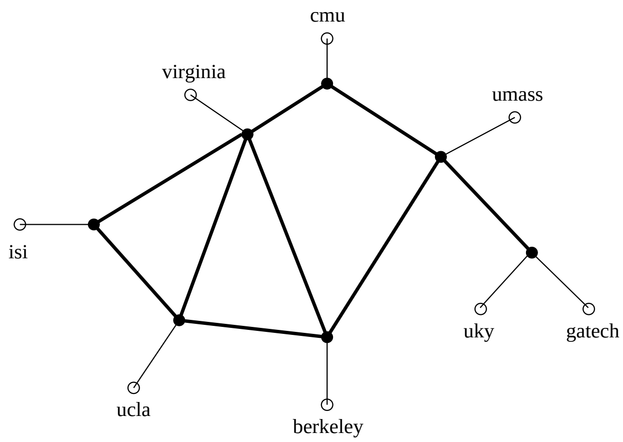


Figure 2: Network topology used in experiments.

overhead incurred by a packet is more difficult to measure as it depends on the route taken by the packet. In addition, the costs of multicast and unicast packets are quite different. In our evaluation, we assume all unicast packets incur same overhead to the network. For a multicast packet that is transmitted to N receivers, we assume it has a cost $N/2$ times the cost of a unicast packet [18]. We call this the *normalized cost* for a multicast packet.

We use an additional performance metric to characterize both the application performance improvement and the protocol overhead. We define the average cost of a recovered packet to be the average number of NACK/repair packets sent and received for each recovered packet.

5.2 Experiments over the MBONE

We modified the MBONE audio tool *vat* to use both STORM and SRM and then ran a series of experiments among 8-12 sites scattered on the MBONE. Figure 2 is a rough approximation of the network topology connecting the 8 hosts involved in the experiments presented here. The topology was generated by finding the multicast route between each pair of sites using the program *mr* [23] and then combining common routes found in *mr*'s output. Figure 3 shows the snapshot of a typical recovery structure created by the hosts during an experiment.

For the SRM implementation, each host in the multicast group exchanges a periodic, low-frequency session message so that the distance between any two hosts can be estimated. We used the following set of parameters in the SRM protocol [9]: $C_1 = C_2 = 2.5$ and $D_1 = D_2 = 2.5$, which, according to our experience, provide acceptable performance.

For both STORM-based *vat* and SRM-based *vat*, we ran many experiments at different times of day, with different hosts as the data source. Due to space limitations, this paper only presents results collected from 6 sets of experiments where each set includes results from one run of STORM and one of SRM. To make sure network conditions are consistent between runs of SRM and STORM, we ran SRM-based *vat*

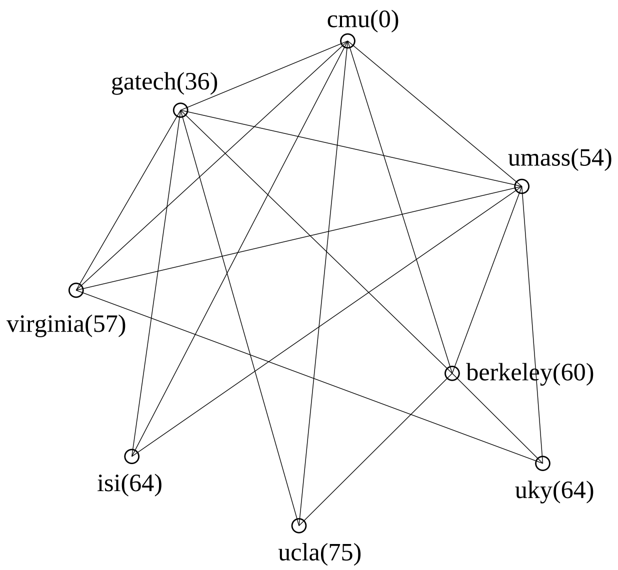


Figure 3: A typical structure built in the experiments. The numbers specify the level assigned to each site. Nodes that are higher in the figure can become parents of lower nodes.

Site	Initial		Final	
	STORM	SRM	STORM	SRM
Berkeley	3.71%	4.11%	0.01%	0.09%
Ga. Tech†	4.37%	4.02%	0.00%	0.29%
ISI	3.82%	3.97%	0.04%	0.11%
UCLA†	3.82%	3.97%	0.35%	0.11%
Kentucky	10.19%	6.88%	0.52%	0.62%
U. Mass†	10.65%	14.46%	0.05%	6.68%
Virginia	42.95%	45.57%	0.17%	22.67%

Table 1: Loss rate (with and without recovery) of all receivers in one of the data sets. Hosts marked with a (†) used a 200 ms playback buffer while unmarked hosts used a 500 ms playback buffer.

right after STORM-based *vat*. The interval between the two experiments was usually less than 10 minutes. We used the same configuration for all experiments: each experiment lasted 5 minutes; a sender at CMU sent PCM-encoded audio in 172 byte packets at a rate of 50 packets per second. All other hosts acted as receivers, some (Ga. Tech, UCLA, U. Mass) with 200 ms playback buffers and the rest with 500 ms playback buffers.

Table 1 shows the loss rate observed by all the receivers in one of our 6 sets of experiments. The columns, marked "Initial" and "Final," show the initial and final loss rates respectively. Tables 2 and 3 show the loss rate seen at receivers Berkeley and U. Mass in all 6 runs of experiments. From these tables we can see the final loss rates achieved by STORM and SRM are usually similar. For sites with a high initial loss rate such as the University of Virginia, STORM

Run	Initial		Final	
	STORM	SRM	STORM	SRM
1	5.73%	5.69%	0.04%	0.17%
2	3.71%	4.11%	0.01%	0.09%
3	1.51%	1.21%	0.01%	0.01%
4	1.83%	1.16%	0.00%	0.00%
5	5.65%	3.88%	0.05%	0.07%
6	1.06%	1.38%	0.00%	0.01%

Table 2: Loss rate seen by receiver Berkeley in 6 runs of experiments.

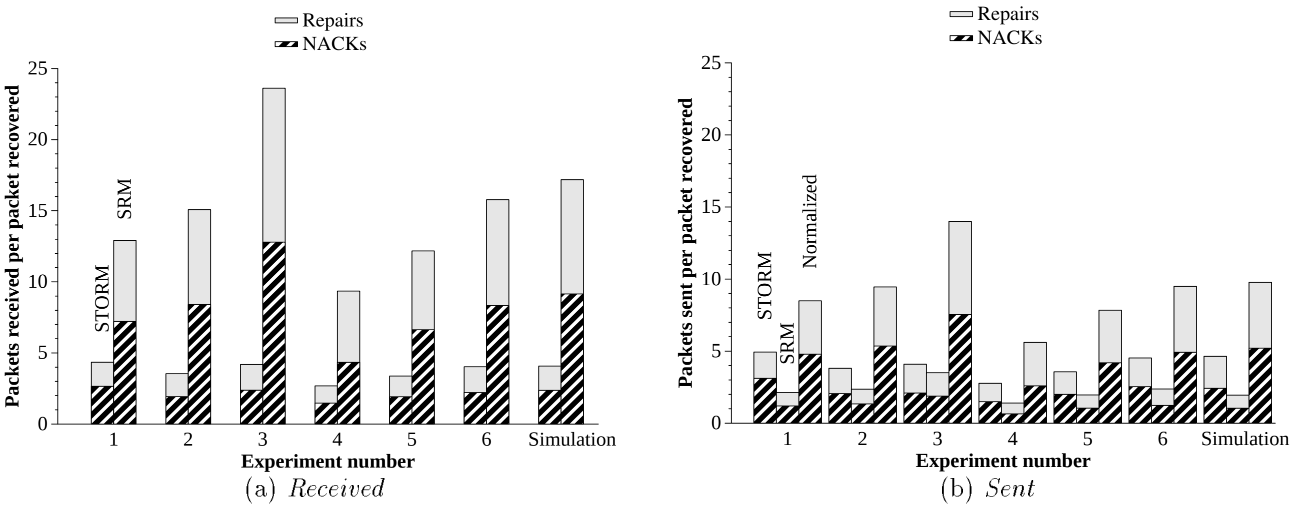
Run	Initial		Final	
	STORM	SRM	STORM	SRM
1	14.16%	11.63%	1.01%	3.35%
2	10.65%	14.46%	0.05%	6.68%
3	5.77%	3.64%	0.22%	1.42%
4	2.60%	2.25%	0.02%	0.88%
5	8.87%	11.60%	0.17%	3.17%
6	4.41%	6.00%	0.20%	2.78%

Table 3: Loss rate seen by receiver U. Mass in 6 runs of experiments.

performs much better than SRM. One explanation is that because MBONE and unicast routes between hosts differ, congestion in the MBONE will not affect unicast packets. Since repair packets in SRM take the same route as original packets, repair packets will experience the same high loss rate. For STORM, repair packets were sent via unicast, avoiding the congestion.

While STORM and SRM provide similar application-level performance, the overheads they incur are quite different. This is illustrated by Figure 4, which depicts the average cost of a recovered packet, the average number of NACKs/repairs sent or received for each packet recovered. The bars marked "Normalized" show the cost of SRM's multicast packets in terms of the cost of unicast packets. The final columns in each graph are data from a simulation of 10 hosts using a randomly generated topology.

There are several noteworthy points about Figure 4. First, in all experiments, the average cost of recovering a packet, both in terms of the number of NACK/repair packets received and the normalized number of NACK/repair packets sent, is much lower for STORM than for SRM. This is a direct consequence of the fact that STORM uses unicast to recover packets from neighboring nodes while SRM uses global multicast for all NACK/repair packets. One benefit of SRM is that it sends fewer packets than STORM. However, since each packet is sent to the whole group, it results in higher network overhead, which is reflected in the normalized number of packets sent and number of packets received. The second thing to notice is that there are certain relationships between the data presented. In particular, the number of packets sent and received for STORM are nearly equal. This is a consequence of using unicast rather than multicast for recovery. Also, the number of packets received in



(a) *Received*

(b) *Sent*

Figure 4: Average number of NACKs and repairs received and sent per recovered packet in 6 experiments and 1 simulation.

SRM is slightly less than half the normalized number of packets sent. This is to be expected given that the number of multicast packets sent is multiplied by $N/2$ to yield the normalized data and that without packet loss, the number of received packets would be N times the number of received packets. Finally, we note that the simulation data matches with the experimentation data closely. The simulator represents a more controlled environment than the MBONE, increasing our confidence in the fact that STORM's lower overhead is not due to differing network conditions between runs of STORM and SRM.

In all the experiments we presented so far, the membership of the multicast group does not change, i.e. all the receivers join the group simultaneously and stay in the group until the end of the session. In order to evaluate the the adaptivity of STORM in the presence of dynamic membership, we performed a number of experiments in which receivers join the multicast group sequentially with certain intervals between joins, stay in the group for a fixed time, and then leave the group. We ran these experiments back to back with experiments without dynamic leave/join. Table 4 compares one typical run of the two experiments. In these experiments all hosts used a 500 ms playback buffer. It can be seen from the table that dynamic joins and leaves do not significantly degrade performance.

Site	Initial		With Recovery	
	Static	Dynamic	Static	Dynamic
Berkeley	1.31%	2.68%	0.0%	0.0%
Ga Tech	1.32%	4.76%	0.0%	0.0%
ISI	4.55%	3.97%	0.0%	0.0%
UCLA	4.55%	4.76%	0.0%	0.0%
Kentucky	3.96%	5.38%	0.59%	1.04%
U. Mass	2.64%	5.50%	0.0%	0.03%
Virginia	3.94%	4.95%	0.0%	0.0%

Table 4: Loss rate of two experiments with STORM, one with static and the other with dynamic group membership. All hosts used a 500 ms playback buffer.

5.3 Simulation Experiments

The main goals of the simulation are to explore how well our protocol scales to a large number of receivers and to evaluate various aspects of the protocol. Before presenting the experimental setup and results, we will first discuss relevant features of the simulator.

We use a discrete event packet level simulator. Identical implementations or code segments of STORM and SRM are used in the experiments on real networks and in the simulator. In the simulator, both unicast and multicast packets are routed along paths that minimize the number of hops. Each link i is characterized by two parameters: a loss rate l_i and a typical delay d_i. For each packet traversing link i, the probability that the packet gets dropped is l_i. If the packet is not dropped, it will be forwarded with a delay that is uniformly distributed between d_i and $2d_i$. With this model, we do not model delay and loss correlations among packets. Furthermore, unlike a real network, the link delay and loss properties are independent of the number of packets traversing the link. The result is that simulations will favor protocols that generate more data. Since SRM with global multicast generates more packets than STORM, the simulator is likely to be overly optimistic about SRM's performance.

Network topologies for use in the simulator are randomly generated. There are two levels of routers in a simulated network: a top level backbone and second level regional networks. Each router at the backbone connects to a second level regional network and each router at the regional network connects to a host that participates in the multicast group. Routers on the backbone are randomly connected to each other such that on average, each router is connected to 4 other

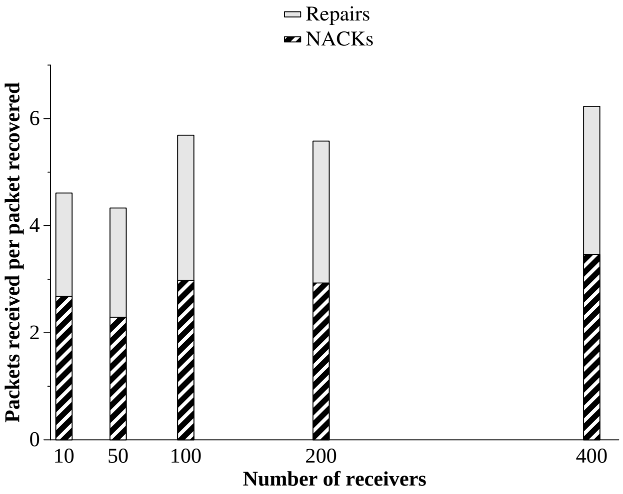


Figure 5: Number of protocol packets received per packet recovered (500 ms playback buffer)

routers. Routers in each regional network are connected in the same way. Backbone links are assigned typical delays on the order of 20 to 40 milliseconds while regional network links are assigned delays of 1 to 5 milliseconds. All links are assigned loss probabilities in the range of 0.1% to 0.5%.

Hosts in the randomly generated topologies tend not to be isolated in contrast to our sites on the Internet, which are all significantly distant from each other. Hosts that are close to each other in the network tend to have a high loss correspondence since they share many links in the multicast distribution tree. Due to the nature of the parent search, it is more likely that a neighbor will be picked as a parent, all other things being equal. When a host loses a packet, the parent to which it sends the NACK is more likely to have also lost the packet. The host needs to wait for its parent to get a repair, increasing the delay to receive a repair slightly and causing more NACKs to be sent in the simulator than in our MBONE experiments due to timeouts.

All hosts joined the multicast group simultaneously at the very beginning of the simulation and remained until the end, 10 minutes later (in simulated time). The effect of all hosts joining the multicast simultaneously is that hosts will have fewer parents to choose from, on average, than if they joined in series since hosts are only able to pick a parent that has already joined the recovery structure. This may lead to a less optimal recovery structure at first, but parent reevaluation and adaptation will subsequently improve the structure.

In order to evaluate how well STORM scales with respect to network size and number of receivers, we ran simulations on topologies with 10, 50, 100, 200, and 400 hosts and with playback buffer sizes of 200 and 500 ms. Figure 5 depicts the average overhead for each host as a function of the group size, where the overhead is measured as the average number of NACK/repair packets received per packet recovered. From the figure, we can see that the cost remains a small constant as the group/network size grows from 10 to 400.

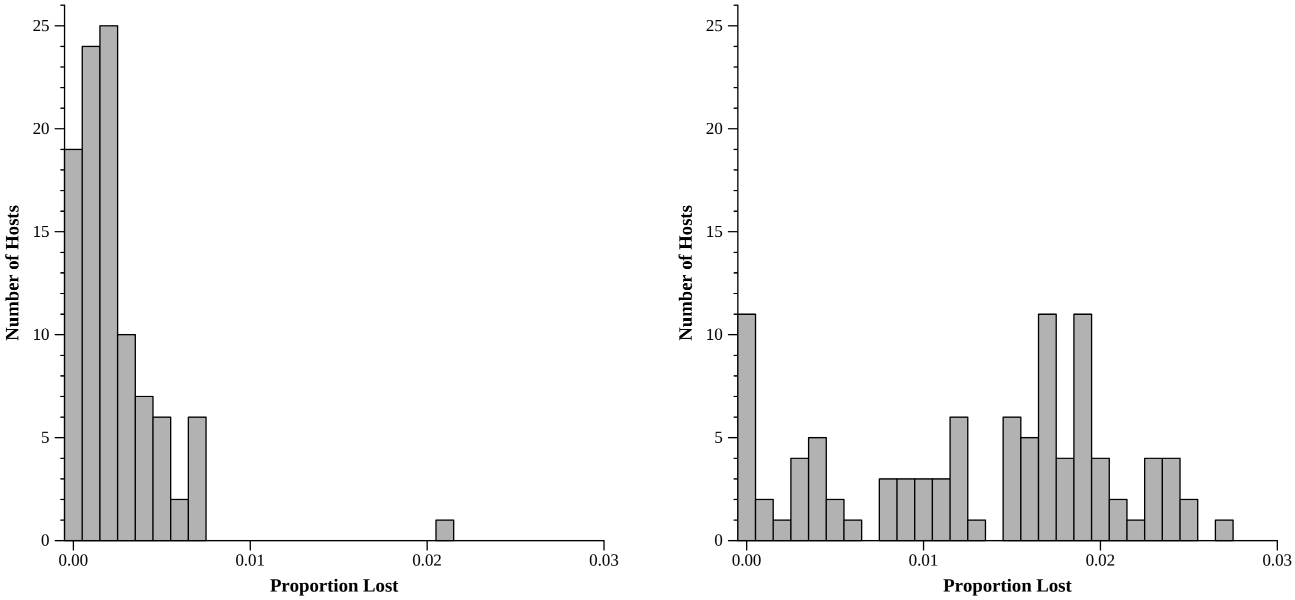
One of the important features of STORM is that the recovery structure is built based on a cost function that takes advantage of the semantics of resilient multicast to choose parents. To evaluate the effect of using the cost function, we ran two simulations of the same 100 host topology using STORM with and without the cost function enabled. We then extracted each host's final loss rate and created a histogram summarizing how many hosts experienced a given level of loss.

Figure 6 presents histograms comparing the frequency of loss for STORM with and without its parent–choosing metric enabled. These histograms were generated using 100 host topologies in which each host used a 200 ms playback buffer. The effect of the metric is particularly obvious in this configuration because the short buffers leave little time for hosts to recover lost packets. The histograms indicate that using the metric does yield a tangible benefit, decreasing the average loss rate from 1.3% to 0.28%. With the metric, it seems that hosts choose parents that are more able to send repairs in time.

6 Related Work

A considerable amount of research has been reported in the area of reliable multicast transport [13, 26, 17]. Instead of describing each approach and comparing it to STORM, we will focus here only on salient similarities and differences between STORM and other approaches in the area of distributed error recovery. Some of the earlier work on reliable multicast transport used a *sender-initiated* approach for error recovery in which the sender is responsible for ensuring that all the receivers receive all the data reliably. However, such an approach does not scale very well as the number of receivers increases. Ramakrishnan and Jain [20] were the first to explore a receiver-initiated approach in which the burden of ensuring reliable delivery is shifted to the receivers. The SRM protocol discussed earlier is an excellent example of this approach in which the receivers are completely responsible to ensure reliable delivery. The tradeoff between sender-initiated and receiver-initiated approaches has been extensively studied [19, 13].

STORM also adopts a receiver-initiated mechanism for error recovery because the resilient multicast model lets each receiver make its own tradeoff between desired latency and the degree of reliability. To facilitate quick, efficient, and robust recovery, STORM also organizes the group members into an acyclic graph so that a receiver can recover a lost packet from one of many parents in the graph. The idea of using a hierarchical error recovery based on a structure among participants is not completely new. TMTP [26] was the first protocol to suggest a tree-based error recovery where the tree is built dynamically among the group members. RMTP [17] is another protocol that uses a statically configured, two-level hierarchy for error recovery. Under TMTP (or RMTP), each receiver has a



(a) *with metric* (b) *without metric*

Figure 6: STORM loss distribution comparing STORM with and without the parent metric (200 ms playback buffer)

single parent and an interior node in the tree is used to aggregate periodic ACKs to ensure feedback on reliable delivery to the sender. Lorax [13] is another protocol that uses a shared tree structure for error recovery in an MxN communication. LBRM [10] is a collection of strategies for achieving receiver-initiated reliable delivery using a hierarchy of logging servers including a primary server responsible for sending positive ACKs to the multicast source.

Another way of building a recovery structure among participants is the Token Ring Protocol (TRP) by Chang and Maxemchuk in which participants are organized into a ring with one of the members acting as the token site. The token site is responsible for timestamping new packets and retransmitting missing packets for all the other receivers. By moving the token site around all the members in the group, the protocol ensures a totally ordered reliable service. Another protocol based on Chang and Maxemchuk's work is RMP (Reliable Multicast Protocol) [24].

The approach used in STORM differs in many ways. First, STORM does not require a parent to maintain any state information (or aggregation of ACKs) about its children. Instead, every receiver identifies potential parents and selects a list of its parents based on each parent's loss rate characteristics. Second, the multi-parent tree structure is built dynamically and adapts as network conditions and loss rates at parents change dynamically. Third, STORM uses a light-weight method for building and maintaining a loop-free structure which keeps the overhead very small. For example, the level-assignment approach allows us to keep the structure loop-free implicitly without explicit exchange of state information among the participants.

As pointed out by Pejhan *et al.* [18], there are three techniques that are used to deal with transmission errors in real-time continuous media applications: automatic repeat request (ARQ), forward error control (FEC), and error concealment (EC). We are using an ARQ mechanism for continuous media applications. Recently, Bolot and others [1] proposed use of FEC for CM applications. One advantage of FEC-based error control is that the recovery latency is smaller than the ARQ method provided the packet transmission rate is high. However, the FEC mechanism usually requires higher bandwidth than the selective retransmission approach. Also, the FEC method requires use of CPU-intensive encoding mechanisms that add to the processing overhead at receivers. Another disadvantage of the FEC approach is that it builds redundancy in transmission assuming homogeneous receiver requirements and loss rates at receivers. Under the resilient multicast model, we assume that receiver requirements are heterogeneous and, therefore, each receiver must be able to make the tradeoff between the overhead of recovery and the degree of desired reliability.

Dempsey and Liebeher [7, 8] and Papadopoulos and Parulkar [16] were the among first to examine the use of retransmission-based error recovery for CM applications. However, their solutions are for unicast sessions and require tighter coordination between the sender and the receiver to recover lost packets. STORM's error recovery scheme is designed for multicast delivery, does not require tight coupling between a sender and its receivers, and relies on a distributed structure to allow recovery of missing packets from one or more (possibly adjacent) members of the multicast group.

7 Conclusion

This paper introduces *resilient multicast*, a new model for multicast delivery of continuous media streams. Under this model, each receiver in a multicast group determines its own tradeoff between desired playback quality and tolerable latency. Based on the tradeoff, a receiver recovers from lost packets by requesting retransmissions from other group members whenever feasible. To facilitate fast recovery and to ensure low retransmission overhead, resilient multicast relies on organizing the participants into a distribution structure (an acyclic graph) for error recovery. We have designed a new multicast delivery protocol called STORM that includes a lightweight algorithm for dynamically creating the distribution structure and a low-cost mechanism for selecting parents for recovery of lost packets.

We have evaluated the efficiency of STORM by comparing it against the SRM protocol and also verified its effectiveness by using both MBONE-based experiments and a simulation model. Our experimental results show that STORM is effective and efficient in error control for real-time multimedia applications. We also show that STORM is adaptive and robust in a real network like MBONE. Further, our simulation results show that STORM has nice scaling properties for large multicast groups.

Acknowledgement

We would like to thank the following people who provided us with remote accounts and helped us with our MBONE experiments: Bruce Mah of University of California at Berkeley, Dan Massey of University of California at Los Angeles, Kevin C. Almeroth and Mostafa Ammar of Georgia Institute of Technology, Ted Faber of Information Sciences Institute, Robert Adams of University of Kentucky, Jim Kurose and Maya Yajnik of University of Massachusetts at Amherst, Deborah Estrin and Pavlin Ivanov Radoslavov of University of Southern California, Paco Hope and Jorg Liebeherr of University of Virginia, Chuck Cranor, John DeHart, Zubin Dittia and Marcel Waldvogel of Washington University at St. Louis, Wieland Holfelder and Rainer Lienhart at University of Mannheim, Germany.

References

[1] J. Bolot, H. Crepin, and A. V. Garcia. Analysis of audio packet loss in the Internet. In *Proceedings of NOSSDAV'95*, pages 163–174, April 1995.

[2] S. Casner. Frequently asked questions (FAQ) on the multicast backbone (MBone), August 1994.

[3] S. Casner and S. Deering. First IETF Internet audiocast. *ACM Computer Communication Review*, July 1992.

[4] J. Chang and N. F. Maxemchuk. Reliable broadcast protocols. *ACM Trans. Computer Systems*, 2(3):251–273, August 1984.

[5] S. Deering. Host extension for IP multicasting, August 1989. RFC-1112.

[6] S. Deering and D. R. Cheriton. Multicast routing in datagram internetworks and extended LANs. *ACM Transactions on Computer Systems*, May 1990.

[7] B. J. Dempsey. *Retransmission-Based Error Control for Continuous Media Traffi c in Packet-Switched Networks.* PhD thesis, Department of Computer Science, University of Virginia, 1994.

[8] B.J. Dempsey, J. Liebeherr, and A.C. Weaver. On retransmission-based error control for continuous media traffic in packet-switching networks. *Computer Networks and ISDN Systems*, 1996.

[9] S. Floyd, V. Jacobson, S. McCanne, C. G. Liu, and L. Zhang. A reliable multicast framework for light-weight sessions and application level framing. In *Proceedings of the ACM SIGCOMM 95*, pages 342–356, Boston, MA, August 1995.

[10] H. Holbrook, S. K. Singhal, and D. R. Cheriton. Log-based receiver-reliable multicast for distributed interactive simulation. In *Proceedings of SIGCOMM'95*, pages 328–341, Boston, MA, August 1995.

[11] V. Jacobson. Multimedia conferencing on the Internet. *SIGCOMM*, August 1994. Tutorial 4.

[12] V. Jacobson and S. McCanne. A visual audio tool.

[13] B. N. Levine, D. B. Lavo, and J. J. Garcia-Luna-Aceves. The case for concurrent reliable multicasting using shared ACK trees. In *Proceedings of ACM Multimedia'96*, November 1996.

[14] J. Lin and S. Paul. RMTP: A reliable multicast transport protocol. In *Proceedings of IEEE INFOCOM'96*, pages 1414–1424, San Francisco, CA, 1996.

[15] S. McCanne and V. Jacobson. vic: A flexible framework for packet video. In *Proceedings of ACM Multimedia'95*, pages 511–522, San Francisco, CA, November 1995.

[16] C. Papadopoulos and G. Parulkar. Retransmission-based error control for continuous media applications. In *Proceedings of the Sixth International Workshop on Network and Operating System Support for Digital Audio and Video*, pages 5–12, 1996.

[17] S. Paul, K. Sabnani, and D. Kristol. Multicast Transport Protocols for High Speed Networks. In *Proceedings of Local Computer Networks*, 1994.

[18] S. Pejhan, M. Schwartz, and D. Anastassiou. Error control using retransmission schemes in multicast transport protocols for real-time media. *IEEE/ACM Transactions on Networking*, 4(3):333–344, June 1996.

[19] S. Pingali, D. Towsley, and J. Kurose. A comparison of sender-initiated and receiver-initiated reliable multicast protocols. In *ACM Sigmetrics'94*, pages 221–230, Nashville, TN, May 1994.

[20] S. Ramakrishnan and B.N. Jain. A Negative Acknowledgment Protocol with Periodic Polling Protocol for Multicast over LANs. In *Proceedings of INFOCOM '87*, pages 502–511, 1987.

[21] H. Schulzrinne. RTP profile for audio and video conferences with minimal control, January 1996. RFC-1890.

[22] H. Schulzrinne, S. Casner, R. Frederick, and V. Jacobson. RTP: A transport protocol for real-time application, January 1996. RFC-1889.

[23] University of Cambridge Computer Lab. http://www.cl.cam.ac.uk/mbone/#Mrouted.

[24] B. Whetten, S. Kaplan, and T. Montgomery. A high performance totally ordered multicast protocol, August 1994. available from research.ivv.nasa.gov as ftp at /pub/doc/RMP/RMP_dagstuhl.ps.

[25] M. Yajnik, J. Kurose, and D. Towsley. Packet loss correlation in the MBone multicast network. In *Proceedings of GLOBECOM'96*, 1996.

[26] R. Yavatkar, J. Griffioen, and M. Sudan. A reliable dissemination protocol for interactive collaborative applications. In *Proceedings of ACM Multimedia'95*, pages 333–344, 1995.