

# Asymptotic Behavior of Global Recovery in SRM

Suchitra Raman, Steven McCanne  
University of California, Berkeley  
{suchi,mccanne}@cs.berkeley.edu

and

Scott Shenker  
Xerox Palo Alto Research Center  
shenker@parc.xerox.com

## Abstract

The development and deployment of a large-scale, wide-area multicast infrastructure in the Internet has enabled a new family of multi-party, collaborative applications. Several of these applications, such as multimedia slide shows, shared whiteboards, and large-scale multi-player games, require *reliable* multicast transport, yet the underlying multicast infrastructure provides only a best-effort delivery service. A difficult challenge in the design of efficient protocols that provide reliable service on top of the best-effort multicast service is to maintain acceptable performance as the protocol scales to very large session sizes distributed across the wide area. The Scalable, Reliable Multicast (SRM) protocol [6] is a receiver-driven scheme based on negative acknowledgments (NACKs) reliable multicast protocol that uses randomized timers to limit the amount of protocol overhead in the face of large multicast groups, but the behavior of SRM at extremely large scales is not well-understood.

In this paper, we use analysis and simulation to investigate the scaling behavior of global loss recovery in SRM. We study the protocol's control-traffic overhead as a function of group size for various topologies and protocol parameters, on a set of simple, representative topologies — the cone (a variant of a clique), the linear chain, and the binary tree. We find that this overhead, as a function of group size, depends strongly on the topology: for the cone, it is always linear; for the chain, it is between constant and logarithmic; and for the tree, it is between constant and linear.

## 1 Introduction

The advent and deployment of IP Multicast [5] has enabled a number of new applications [17, 10, 9, 7, 22] that utilize large-scale multipoint communication over wide-area networks. IP Multicast extends the traditional, best-effort unicast delivery model of the Internet architecture to enable efficient multipoint packet delivery. In this model, the network delivers a packet from a source to an arbitrary number of receivers by forwarding a copy of that packet along each link of a distribution tree rooted at the source subnet (or, de-

pending on the routing protocol, at a rendezvous point [4] or core router [1]). As with unicast, IP multicast is not *reliable* — packets might be dropped at any point along the distribution tree. However, many new multicast applications like shared whiteboards, webcast tools, and distributed simulation are not tolerant of packet losses. Whiteboard state, for example, is persistent; if a piece of a drawing update is lost, the application cannot leave the drawing in an incomplete state. Instead, that application must recover the missing packet to repair the damaged portion of the drawing. In short, this particular application, and in fact a large class of emerging applications, require a *reliable multicast* transport protocol. Although mechanisms for reliable unicast transmission are comparatively well-understood and have proven extremely successful (*e.g.*, TCP), making multicast reliable at large scales remains a formidable challenge.

A fundamental problem in the design of a reliable multicast protocol is the well-known *message implosion* [6, 19] problem. Reliable transport protocols rely on some form of feedback between or among communicating end-points to confirm the successful delivery of data. While some protocols rely on positive acknowledgments or ACKs (signalling the successful receipt of data), others rely on negative acknowledgments or NACKs (signalling the failure to receive expected or desired data). Positive acknowledgment-based schemes are successful for reliable unicast transport but scale poorly in the multicast case when there are many receivers. In this case, each delivered packet causes a flood of positive acknowledgments sent from the receivers back to the source, overwhelming either the source or the intervening routers, if not both.

A number of solutions to the ACK implosion problem have been proposed. Log-based reliable multicast [8] uses logging servers to constrain recovery traffic to localized groups of receivers. TMTP [24] and Lorax [12] construct a hierarchy in the form of a tree, in which multiple identical ACKs are fused together before they are propagated up the tree toward the root. RMTP [13] uses a similar approach based on trees that are (statically or dynamically) configured into the network rather than constructed by the application. XTP [2] takes a markedly different approach, however, and instead multicasts control traffic to all end-points. To limit the proliferation of this control traffic, XTP employs a “slotting and damping” algorithm: a receiver waits for a random amount of time before generating control traffic and cancels that message if some other hosts multicasts the same information first. The algorithms in SRM [6] elaborate this simple yet powerful primitive with adaptive timers that improve performance across wide-area, heterogeneous networks.

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. To copy otherwise, to republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee. SIGMETRICS '98 Madison, WI USA © 1998 ACM 0-89791-982-3/98/0006...\$5.00

While TMTP, Lorax, and RMTP limit recovery traffic using unicast transmission over an artificially constructed hierarchy, XTP and SRM limit recovery traffic using multicast transmission and explicit suppression. Although this latter approach is potentially more robust because it does not require an elaborate protocol for tree construction, maintenance, and reconfiguration, it also entails potentially more overhead because recovery traffic is multicast to the entire group and not just to those members impacted by the packet loss. To address this problem, [6] proposes that their SRM reliable multicast framework be cast as two complementary pieces: a *global recovery* component that ensures the delivery of all desired data across the entire multicast session, and a *local recovery* component that constrains the reach of recovery traffic to the multicast neighborhoods where packet loss occurs. Although [6] focuses primarily on global recovery, the SRM authors argue that local recovery is an important and necessary optimization to scale their protocol to large, heterogeneous sessions. Since then, several promising approaches to local recovery have been proposed [11, 14] and the problem remains a focal point of ongoing research.

Even though a viable local recovery strategy is critical to SRM's scalability, in certain configurations (e.g., where packet loss occurs near the root of the distribution tree), the degree to which local recovery enhances performance may be limited and the protocol's overall performance may strongly depend on that of the global recovery scheme. Hence, we claim that a thorough understanding of global recovery in SRM is not only important in and of itself, but will also be useful in predicting the performance of SRM even when coupled with local recovery.

In this paper, we use analysis and simulation to investigate the scaling behavior of global loss recovery in SRM. We study the growth control traffic (measured by NACK counts) as a function of group size for various topologies and protocol parameters, on a set of simple, representative topologies — the cone, the linear chain, and the binary tree. We find that the number of NACKs, as a function of group size, for the cone is always linear, for the linear chain is between constant and logarithmic, and for the tree is between constant and linear.

The rest of this paper is organized as follows. We start with a brief overview of the SRM protocol in Section 2. Section 3 summarizes related work. In Section 4, we describe our simulation methodology. We discuss the effects of varying the protocol parameters for the various topologies in Sections 5, 6, and 7, and conclude in section 8.

## 2 Overview of SRM

SRM is a NACK-based, fully-decentralized reliable multicast protocol originally described by Floyd, *et al.*, in [6]. The SRM framework builds on Clark and Tennenhouse's principle of Application Level Framing (ALF) [3], which provides an elegant solution to the problem of reliable-multicast API design because its flexibility offers applications the opportunity to actively participate in the loss-recovery procedure.

To avoid ACK-implosion, SRM uses NACKs. Receivers detect losses from discontinuities in sequence numbers (or by other means with a generic data naming scheme [20]) and transmit NACKs as a request for retransmission of the lost data<sup>1</sup>. A randomized algorithm determines when a receiver

transmits a NACK. These NACKs are multicast to the entire group so that any receiver, in particular the closest receiver with the requested data, may generate a repair in response to a NACK. The repair messages are also *multicast* to the entire group, so that all receivers that missed that packet can be repaired by a single response. The repair message traffic likewise makes use of the randomized timer algorithm.

To avoid NACK implosion, receivers that observe a NACK for data that they too have not received do not send their own NACK<sup>2</sup> and await the repair data. The goal of the randomized NACK transmission algorithm is to minimize the number of duplicate NACK messages sent. To accomplish this, each receiver delays the transmission of a NACK by an amount of time given by the expression

$$\text{backoff} = D \cdot (C_1 + C_2 r)$$

where *backoff* is the amount of delay, *D* is an estimate of the one-way delay from the receiver to the source that generated the lost data packet, *C*<sub>1</sub>, *C*<sub>2</sub> are non-negative protocol constants, and *r* is a uniformly distributed random number in [0, 1]. This random delay provides receivers with the opportunity to *suppress* the transmission of similar pending NACKs; that is, delaying the transmission of NACKs by a random amount increases the likelihood that a NACK from one receiver is delivered to another receiver before that receiver sends its own NACK, and thus, reduces the total number of NACKs. Figure 1 illustrates the suppression mechanism in SRM.

As in [6], we call *C*<sub>1</sub>*D* the *deterministic* delay and *C*<sub>2</sub>*Dr* the *random* delay. The deterministic-delay component induces suppression effects across receivers situated at varying distances from the point of loss (e.g., a chain topology), while the random-delay component induces suppression effects across receivers situated at equal distances from the point of loss (e.g., a star topology). We say that a receiver's timer *fires* if no suppressing NACK has been received when its backoff period has expired.

Since NACKs are multicast to the group, any receiver that has the data can respond, not just the original source. However, we again have the potential for a control-traffic storm if all hosts respond simultaneously. Thus, to avoid repair-packet storms, SRM reuses its NACK suppression machinery to limit the number of redundant repair packets. Because both NACKs and repairs are sent to the entire multicast group, we call this the SRM *global recovery* mechanism.

A number of performance metrics have been used to characterize recovery schemes for reliable multicast, but two widely used metrics are:<sup>3</sup> (1) the degree of duplicate control traffic, and (2) the recovery latency. The first metric can be summarized as the average number of NACKs sent for each dropped packet, which clearly depends on the size of the group experiencing the loss. We denote this number by *N*(*G*), where *G* is the number of members experiencing the loss. The larger this metric, the less effective the randomized timer algorithm is at suppressing duplicate NACKs and

SRM is receiver-reliable and does not require that all receivers obtain all data. Instead, receivers issue "repair requests" to repair only those data wanted. For this paper, we use the terms "NACK" and "repair request" interchangeably.

<sup>2</sup>More precisely, they scale their transmission timer awaiting a response. All receivers, if they have not received the repair data, will eventually transmit a NACK.

<sup>3</sup>The metrics we describe here ignore topological heterogeneity, where not all receivers are identical. More detailed performance metrics would measure the latencies on a per receiver basis.

<sup>1</sup>To be true to the original intentions of the SRM designers, we must admit that our use of the term "NACK" is somewhat inaccurate since it implies that the underlying protocol generates NACKs to guarantee that all data is eventually received by all receivers. In fact,



that exponentially distributed timers yield better scaling properties. They found that having this distribution depend on the group size could result in improved scaling. We do not address the effects of different timer distributions at any great length in this paper.

This paper is primarily concerned with global recovery in SRM with constant  $C_1$  and  $C_2$ . Variants of SRM have been proposed that use local recovery, in which NACKs and repairs are not sent to the entire group. [6], [14] look at two methods to limit the range of these methods: hop-scoping, and local recovery groups. [15] considers methods for adaptively setting the values for  $C_1$  and  $C_2$ . We do not consider any of the local recovery methods, nor adaptive timer setting. Thus, our work should not be seen as a statement about how SRM-like protocols should function in the future, when they may well incorporate such features, but rather as an attempt to study the current deployed version of SRM with its use of global recovery. Our hope is that understanding this basic version of the protocol may inform future design efforts to improve it.

#### 4 Simulation Methodology

In our simulations, we studied three classes of network topologies: *cone*, *linear chain*, and *binary tree*, each with a single source. The cone is a topology where each member has the same delay  $\delta$  to every other member, and a distance  $\Delta$  from the source. Similarly, for the linear chain and the binary tree,  $\delta$  represents the link delay between adjacent members, and  $\Delta$  is the link delay from the source to the closest member(s). Figures 2 and 3 show  $\Delta$  and  $\delta$  for the three topologies.

We are only modelling the behavior of NACKs, so we need only consider the receivers that suffer losses. Thus, we only consider the case where the loss occurs on the link adjacent to the source<sup>5</sup>. This causes little loss of generality, since if the loss occurs elsewhere we need only model the topology beneath the loss point. Note, however, that then the size of the group we are considering,  $G$ , is the size of the *loss group* – the number of members experiencing a particular packet loss – and not always the size of the entire group. Session messages in SRM give members knowledge about the size of the entire group, but not about the size of the loss group. If members knew the size of the loss group they might also be able to employ various forms of local recovery (hop-scoped recovery, or local recovery groups) that would more directly address the NACK traffic problem (not just limiting the number of NACKs, but also the portion of the group they are sent to). Thus, we do not consider varying the timer constants with group size, as in [18], as this does not seem like a realistic possibility.

Furthermore, we assume that losses are detected immediately when the next packet arrives. Since a packet is delivered to different receivers at different absolute times, losses are detected at different times. This typically allows the receivers closer to the source to suppress the NACKs from receivers further away. One of the key points in our investigation is how the setting of the timer constants affects this behavior.

We used the VINT network simulator *ns* [16] for our work. In its original form, *ns* turned out to have prolific memory usage with heavy-weight nodes, links, and multicast routing infrastructure, and could not support more than a few hundred nodes on an ordinary workstation. However,

<sup>5</sup>Measurements reported in [23] show that most correlated losses occur close to the source.

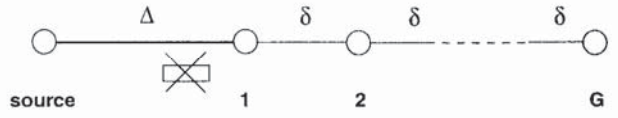


Figure 2: Linear Chain Topology: the X-ed packet marks the location of packet loss.

we took advantage of *ns*'s extensible object-oriented architecture and made several modifications and extensions to it. Using the basic *ns* framework for event handling, we extended the simulator to support regular topologies with static routing without explicit routing table state. These modifications and extensions to *ns* enabled large-scale simulations of up to 50,000 nodes.

Losses occur on the link closest to the source, and are thus shared by all receivers in the group. We measure the average number of NACKs generated in response to a loss. The variation between different measurements is induced by the randomness in the recovery algorithm we are studying. We ran between 30 and 50 simulations of each case to compute the average value of the metrics, depending on the variance of the measured samples. Table 1 summarizes notation used in the rest of this paper.

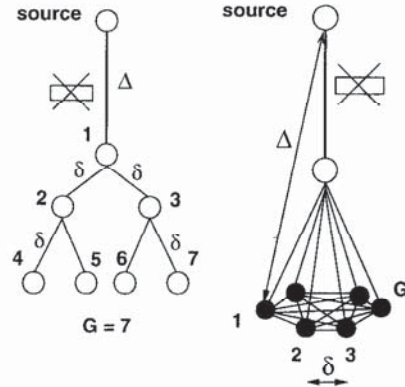


Figure 3: Binary tree and cone topologies: the X-ed packet marks the location of packet loss.

Symbol	Description
$\Delta$	Delay from source to the closest receiver
$\delta$	Delay of link connecting receivers
$R$	$\Delta/\delta$
$G$	Group size
$N$	Average number of copies of a single NACK
$L$	Average NACK latency caused by backoff
$D_i$	Estimate of one-way delay from node $i$ to the source node
backoff $_i$ , at host $i$	$D_i \times (C_1 + C_2 \times r_i)$ where, $r_i$ are uniformly distributed random variables in $[0, 1]$
$t_i$	Absolute time at which receiver $i$ 's timer fires

Table 1: Summary of notation

In the following sections, we present our analytical and

simulation results for the three topologies: cone, line and binary tree.

## 5 Scaling in the Cone Topology

The cone topology can be used to model the case of a broadcast LAN. If the source is on the LAN then  $\Delta = \delta$  but when the source is off the LAN, the delay from the LAN to the source is much greater than the LAN propagation time, yielding  $\Delta \gg \delta$ . In general, the cone can be used to model a topology where all receivers have similar round-trip time estimates to the source. In practice, RTT estimators tend to be coarse-grained resulting in clusters of receivers with similar RTT values.

We use the following probabilistic analysis to compute the expectation of  $N(G)$ . Because all the receivers are at the same distance from the loss in a cone, the deterministic backoff component has no impact on the number of duplicates (all timers have the same constant offset). The average delay in transmitting the first NACK depends on the expected value of the minimum timer and is given by  $\Delta(C_1 + \frac{C_2}{G+1})$ . This result follows directly from noting that the expectation of the minimum of  $G$  uniformly distributed random variables in  $[0, 1]$  is  $\frac{1}{G+1}$ . The number of duplicates is equal to the expected number of timers that fire within  $[t_{min}, t_{min} + \delta]$ , where  $t_{min}$  is the value of the smallest timer. Since backoffs are uniformly distributed in  $[C_1\Delta, (C_1 + C_2)\Delta]$ , we can easily compute this expectation. Defining  $\alpha = \frac{\delta}{C_2\Delta}$  we have,

$$E[N] = \begin{cases} 1 + G\alpha - \alpha^G, & \alpha < 1 \\ G, & \alpha \geq 1 \end{cases}$$

Thus, the number of duplicates is roughly linear in the group size. [6] reports a similar result for the *star* topology, which is a *cone* with  $\Delta = \delta$ . Observe that this linear dependence applies regardless of whether the delay estimates are accurate or not. If the estimated value of the delay (assuming all members achieve the same estimate) is larger than the true estimate, then the number of duplicates is smaller, but the dependence on  $G$  is still linear. Our simulations, shown in Figure 4, confirm this result.

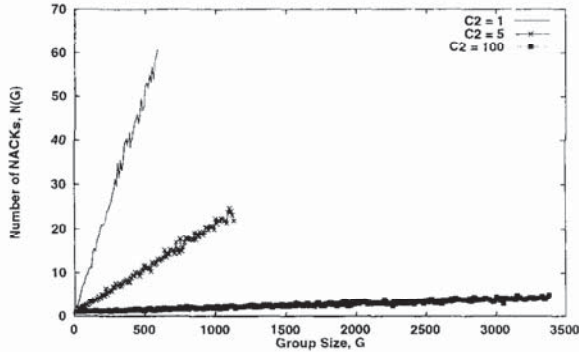


Figure 4: In the cone topology,  $N(G)$  grows linearly in  $G$ , where  $\alpha = \frac{\delta}{C_2\Delta}$  and  $C_1 \geq 0$ ,  $C_2 > 0$ .

$N(G)$  grows roughly linearly for any fixed timer distribution. However, as shown by Nonnenmacher and Biersack [18], if one makes the distribution dependent on the size of

the loss group then one can change this linear scaling. For instance, if one takes a bimodal distribution such that with a probability  $p = \frac{a}{G}$  a receiver sends a NACK immediately upon detecting a loss, and with probability  $1 - p$  sends a NACK after a delay  $\delta$ , then as  $G$  diverges  $N(G)$  is given by  $a(1 - e^{-a}) + Ge^{-a}$ . By tuning  $a$  one can lower the slope of the linear dependence, and if one sets  $a = \ln G$  the growth is logarithmic, not linear. One can remove the linear term entirely by considering the scheme where each receiver picks a number  $k$  from an exponential distribution with average  $\frac{a}{G}$  and sets the backoff to  $k\delta$ . This is essentially a discrete version of the exponential distribution considered by Nonnenmacher and Biersack [18]. Here, the average number of NACKs is  $E(N) = a$  and the average latency is  $E(L) = \frac{e^a \delta}{1 - e^{-a}}$ . One can show that this achieves the lowest latency for a given number of NACKs (or equivalently, the smallest number of NACKs for a given latency) in the asymptotic limit. However, as we argued earlier, schemes that have the timer distribution depending on  $G$  are perhaps of little interest since the parameter  $G$  must be the size of the loss group, and once one has this information it might be better used in some local recovery approach rather than using it merely to tune the timer parameters.

## 6 Linear Chain

For the linear chain topology, we first consider the case where RTT estimation is exact. When  $C_1 > 0$  and  $C_2 \geq 0$ , the data in Figure 5 suggests that  $N(G)$  is constant in  $G$ .

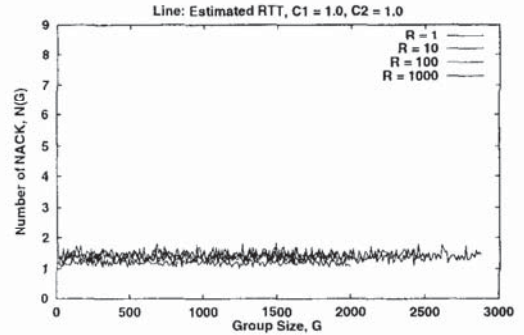


Figure 5:  $N(G)$  is a constant for  $\Delta/\delta = 1, 10, 100, 1000$ , with exact RTT estimation and  $C_1 = C_2 = 1$ . Similar results hold for other  $C_1$  and  $C_2$  as long as  $C_1 > 0$ .

We now show that in this parameter range there is a bound  $k$  on the maximal number of NACKs sent. Receiver  $i$  picks  $C_1(\Delta + (i-1)\delta) \leq \text{backoff}_i \leq (C_1 + C_2)(\Delta + (i-1)\delta)$ . Consider some message sent at time  $t = 0$ , and assume that losses are detected immediately. Receiver  $i$  detects the loss at time  $(\Delta + (i-1)\delta)$  and sends its NACK (if not suppressed) no later than a time  $(\Delta + (i-1)\delta) + (C_1 + C_2)(\Delta + (i-1)\delta)$  and no sooner than  $(\Delta + (i-1)\delta) + C_1(\Delta + (i-1)\delta)$ . Therefore, receiver  $i$  and receiver  $j$  cannot both send NACKs if, assuming  $j > i$ ,

$$(C_1 + C_2)(\Delta + (i-1)\delta) + (j-i)\delta < (j-i)\delta + \delta + C_1(\Delta + (j-1)\delta)$$

This follows by recalling that it takes time  $(j-i)\delta$  for  $i$ 's NACK to propagate from  $i$  to  $j$ . Thus, the first member

on the line suppresses all but the next  $k$  members, where  $k$  is given by  $k = \lfloor \frac{C_2 \Delta}{C_1 \delta} \rfloor$ . Thus,  $N(G)$  is bounded above by  $k + 1$ . The simulations suggest that the *average number*  $N(G)$  is much less than this upper bound, and in particular, is independent of  $R$ .

For  $C_1 > 0$ , the value of  $N(G)$  appears, as shown in Figure 6, to be roughly independent of  $C_2$ . The dependence on  $C_1$  is also shown in Figure 7, where, for a fixed  $G$ ,  $N$  decreases with increasing  $C_1$  as expected.

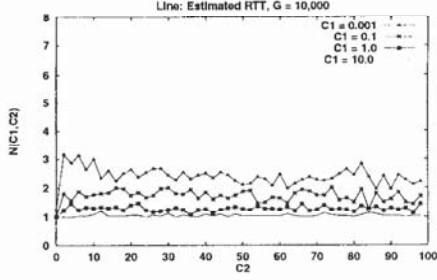


Figure 6:  $N$  as a function of  $C_1$  and  $C_2$ .

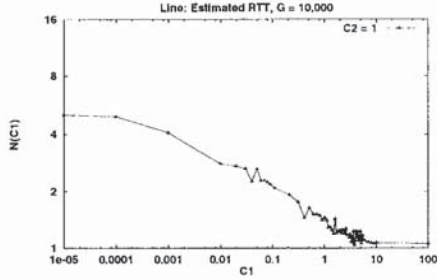


Figure 7:  $N$  as a function of  $C_1$ .

When  $C_1 = 0$ , there is no deterministic delay and the preceding argument fails. In fact, it appears that  $N(G)$  diverges slowly with the group size  $G$ , as shown in Figure 8. We can argue that  $N(G)$  does not grow faster than a certain expression derived below (but are not able to provide a lower bound). The probability that node  $i$  is not suppressed is bounded above by the probability that it is not suppressed by the members ahead of it in line. This occurs if and only if (ignoring ties) the backoff timer  $t_i = \min\{t_1, \dots, t_{i-1}\}$ . Considering the case  $\Delta = \delta$  for convenience. Using the notation  $z_+ = \max[0, z]$ , we have

$$Pr[t_i = \min\{t_1, \dots, t_{i-1}\} | t_i = x] = \prod_{j=1}^{j=i-1} Pr[t_j \geq x]$$

$$= \prod_{j=1}^{j=i-1} (1 - x/j\delta)_+$$

and so, changing variables,

$$N(G) \leq \sum_{i=1}^G \int_0^1 \prod_{j=1}^{j=i-1} (1 - \frac{y}{j}) \frac{dy}{i}$$

Approximating  $\prod_{j=1}^{j=i-1} (1 - \frac{y}{j})$  as  $e^{-y \sum_{j=1}^{i-1} \frac{1}{j}}$  and then noting that  $e^{-y \sum_{j=1}^{i-1} \frac{1}{j}} \approx e^{-y \ln i}$  and substituting into the integral, we see that this expression diverges as  $\ln \ln G$ .

We now consider the case where there is no RTT estimation, and all receivers use the same hardwired delay estimate  $D$ . Note that since deterministic delay is useless when round-trip times are not used (all members have the same deterministic delay),  $C_2 = 0$  results in no suppression at all, and  $N(G) = G$ . This is true independent of topology; if there is no RTT estimation, then one needs  $C_2 > 0$  or else  $N(G) = G$ , and  $N(G)$  is independent of  $C_1$ .

Figure 9 shows  $N(G)$  for the case  $C_1 = 0$  and  $C_2 = 1$  and fixed RTT. The growth, for all values of  $R = \frac{\Delta}{\delta}$  appears to be logarithmic. Similar logarithmic-like behavior is observed in simulations with different values for  $C_2$  and  $D$ .

The following probabilistic analysis suggests why, for  $C_1 = 0$  and  $C_2 = 1$ ,  $N(G)$  grows as a logarithmic function of the group size. The backoffs are picked in the range  $[0, D]$ . We first compute the probability that the NACK at node  $i$  is not suppressed. The following condition must hold, for  $i$ 's timer to fire:

$$d_j + r_j \delta + d_{ji} \geq d_i + r_i \delta, \forall j \neq i$$

where  $d_j$  is the one-way delay to receiver  $j$  from the source and  $d_{ij}$  is the one-way delay from receiver  $i$  to receiver  $j$ .  $r_i, r_j$  are uniformly distributed random numbers picked in  $[0, 1]$  by the random timer mechanism. We then must have:

$$r_i < r_j, \forall j < i \quad (1)$$

$$r_j \delta + 2d_{ij} > r_i \delta, \forall j > i, \text{ and} \quad (2)$$

$$d_{ij} \geq \delta, \forall d_{ij} \quad (3)$$

$$\Rightarrow r_i \delta < 2\delta + r_j \delta, \forall j > i \quad (4)$$

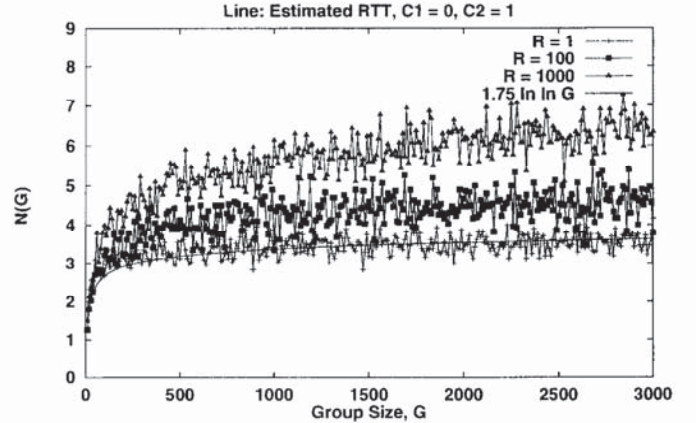


Figure 8:  $N(G)$  diverges as  $\Delta/\delta = 1, 10, 100, 1000$ , with RTT estimation,  $C_1 = 0$ ,  $C_2 = 1$ .

From equation 4 above, we can conclude that a NACK at node  $i$  cannot be suppressed by a NACK at a later node. The condition for suppression at node  $i$  is therefore  $r_i \leq \min\{r_1, r_2, r_3, \dots, r_{i-1}\}$ . Thus,  $P[i \text{ fires}] = \frac{1}{i}$  and so  $E[N] = \sum_{i=1}^G P[i \text{ fires}] \approx \ln G + 0.577$ . Similar logarithmic growth is seen empirically for larger  $C_2$ . The behavior of  $N(G)$  for the line case is summarized in Table 6.

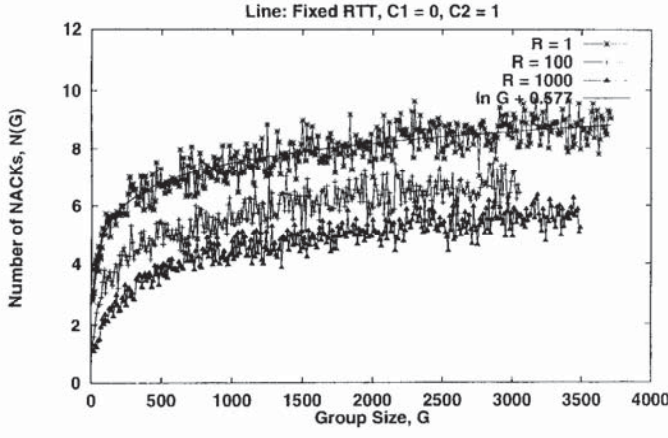


Figure 9:  $N(G)$  grows as a logarithmic function of  $G$  for  $\Delta/\delta = 1, 10, 100, 1000$ , fixed delay (no RTT estimation),  $C_1 = 0, C_2 > 0$ .  $N(G) = \ln G + 0.577$ , when  $C_1 = 0, C_2 = 1$ .

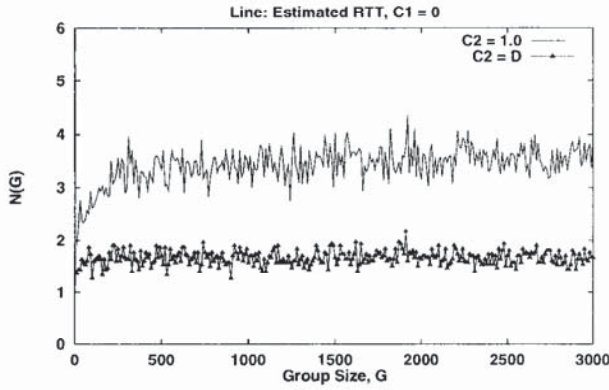


Figure 10:  $N(G)$  converges to a constant when  $C_2 = \sqrt{D}$  for the linear chain.

With  $C_1 = 0$ ,  $N(G)$  grows as  $\ln \ln G$  for the linear chain topology. In order to reduce this growth in  $N(G)$  to a constant, while still retaining  $C_1 = 0$  for the sake of low latency, we can make  $C_2$  a function of the delay from the source. This follows the work Liu *et al.* who propose, in [15], using a new adaptive timer algorithm. Analysis similar to the previous case (equation (4)) shows that the number of duplicates is bounded by a constant when we use  $C_2 = D^\epsilon$  for any  $\epsilon > 0$ . This is because

$$\begin{aligned} N(G) &\leq \sum_{i=1}^G \int_0^1 \prod_{j=1}^{j=i-1} \left(1 - \frac{y}{j^{1+\epsilon}}\right) \frac{dy}{i^{1+\epsilon}} \\ &\leq \sum_{i=1}^G \frac{1}{i^{1+\epsilon}} \leq \frac{1}{\epsilon} \end{aligned}$$

The graph in Figure 10 shows that  $N(G)$  converges to a constant for  $\epsilon = 0.5$ . We should note that because we do not have a lower bound for the case of  $C_2$  fixed ( $\epsilon = 0$ ). Our simulation results show that  $N(G)$  diverges for  $\epsilon = 0$ , but our analytical proof is only for  $\epsilon > 0$ .

## 7 Binary Tree

In the binary tree topology (Figure 3),  $N(G)$  grows linearly with  $G$  when RTT is not estimated, as shown in Figures 11 and 12. The slope of this linear growth depends on  $C_2$  and  $D$  (the fixed RTT). This linear behavior is in contrast with the logarithmic behavior observed in the line topology, but similar to the behavior in the cone topology. When RTT is known exactly, we still have linear behavior for  $C_1 = 0$ , as shown in Figure 12. The slope of this linear growth depends on both  $\frac{\Delta}{\delta}$  and  $C_2$ .

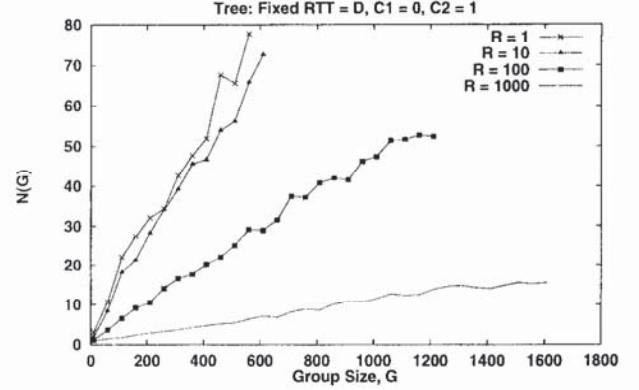


Figure 11: With  $C_1 = 0, C_2 > 0$  and without RTT estimation,  $N(G)$  scales linearly with  $G$  for different values of  $R = \Delta/\delta$ .

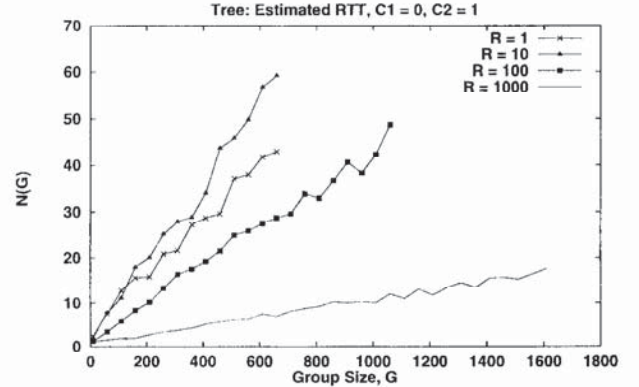


Figure 12: With  $C_1 = 0, C_2 > 0$  and with accurate RTT estimation,  $N(G)$  scales linearly with  $G$  for different values of  $R = \Delta/\delta$ .

However, as soon as we have  $C_1 > 0$ ,  $D(G)$  appears to asymptotically reach a constant. Figure 13 shows the function  $N(G)$  for different values of  $0 \leq C_1 \leq 1$ . The growth law for intermediate  $G$  is linear, and then the slope decreases as  $G$  increases. For all cases where we have been able to reach sufficiently large  $G$ , the slope continues to decrease until  $N(G)$  goes to a constant.

When  $C_1 > 0$ , we see that the asymptotic scaling behavior depends on whether deterministic suppression or randomized suppression is dominant in reducing the number of

$\Delta/\delta$	RTT	$C_1$	$C_2$	$N(G)$	Figure
1, 10, 100, 1000	Fixed	$C_1 \geq 0$	$C_2 > 0$	Logarithmic ( $\ln G + \gamma$ , when $C_1 = 0, C_2 = 1$ )	9
1, 10, 100, 1000	Fixed	$C_1 > 0$	$C_2 = 0$	Linear ( $N(G) = G$ )	
1, 10, 100, 1000	Estimated	$C_1 > 0$	$C_2 \geq 0$	Constant ( $\leq 4$ )	5
1, 10, 100, 1000	Estimated	$C_1 = 0$	$C_2 > 0$	Diverges	12

Table 2: Scaling behavior in the linear chain topology

NACKs. In cases where deterministic suppression is dominant, the asymptotic scaling is constant. Scaling is linear when suppression depends on the randomized suppression. In Figure 16, these two important effects are evident: as  $\Delta/\delta$  increases, deterministic suppression becomes weaker and randomized suppression is more effective. For large values of  $\Delta/\delta > 100$ , backoff timer ranges are large enough and the average separation between timers grows.

We now try to illustrate this behavior in a different form. The function  $\frac{G}{N}$  plotted against  $G$  is shown in Figure 14. This ratio appears to be a linear functions of  $G$ , with the slope depending on  $C_1$ . If we label the slope of this line by  $m$  and the intercept by  $f$ , we have, for small  $C_1$  and large  $G$ , the following form for  $N$ :

$$N = \frac{G}{mG + f}$$

The fit parameters  $m$  and  $f$  are functions of  $C_1$  and  $C_2$ . This linear fit applies over a wide range of  $C_1, C_2$  values. This functional form for  $N(G)$  is consistent with our observation of a linear increase for small values of  $G$ , followed by this slope decreasing and the curve flattening to a constant. In particular, note that  $\lim_{G \rightarrow \infty} N \rightarrow \frac{1}{m}$ , a constant for a given value of  $C_1$  and  $C_2$ . Thus, the slope of this functional fit in Figure 14 yields the asymptotic value for  $N(G)$ . Figure 15 shows this dependence on a log scale.  $\frac{1}{m}$  decreases with increasing  $C_1$  as expected.

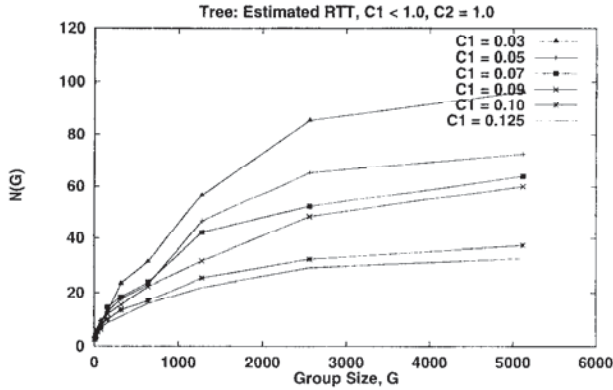


Figure 13:  $N(G)$  in the binary tree for  $R = \Delta/\delta = 1$ , accurately estimated RTT and  $0 < C_1 \leq 1$ ,  $C_2 = 1$ .

If we hold  $G$  fixed and vary  $R$  (the ratio of  $\Delta$  to  $\delta$ ) we find that the dependence is not monotonic. Figure 17 shows this unimodal behavior. This behavior may be explained by the following reasoning. There are two kinds of suppression, deterministic and random, so-called depending on whether the possible firing times overlap or not. Deterministic suppression decreases with  $R$ , but random suppression

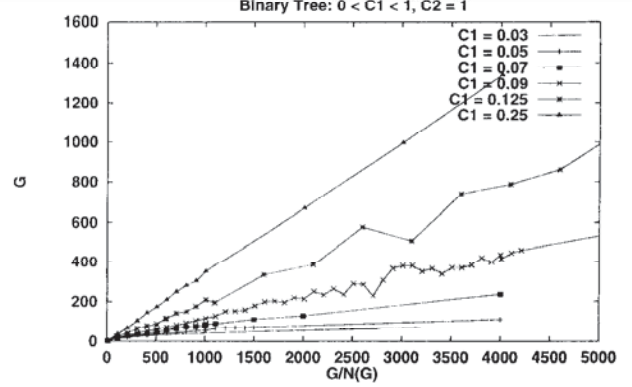


Figure 14:  $G/N$  vs.  $G$  in the binary tree for  $R = \Delta/\delta = 1$ , RTT estimated,  $0 < C_1 \leq 1$ ,  $C_2 = 1$ .

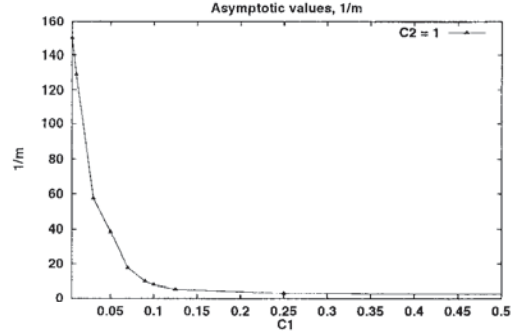


Figure 15:  $\frac{1}{m} = \lim_{G \rightarrow \infty} D(G, C_1)$  as  $C_1$  is varied.  $C_2 = 1$

increases with  $R$ . Thus, as  $R$  is increased we first see an increase as the deterministic suppression becomes less effective, and then see a decrease as random suppression becomes dominant and deterministic suppression is no longer much of a factor (and so cannot decrease significantly further).

With  $C_1 = 0$ , and  $C_2 > 0$ ,  $N(G)$  grows linearly with  $G$ . In order to reduce this growth in  $N(G)$  to a constant, while still retaining  $C_1 = 0$ , as we did for the linear chain topology, we make  $C_2$  a function of the delay from the source. The adaptation algorithm described in [15] results in  $C_2$  values that increase roughly linearly in  $D$ , the distance of a receiver from the source.

Here we do not model the dynamics of the adaptation, but instead merely insert the dependence on  $D$  directly. We consider several variants, with  $C_2$  increasing as  $D$ ,  $D^2$ , and  $\sqrt{D}$ . Figure 18 shows the results of these simulations. We

$\Delta/\delta$	RTT	$C_1$	$C_2$	$N(G)$	Figure
1, 10, 100, 1000	Fixed	$C_1 \geq 0$	$C_2 > 0$	Linear	12
1, 10, 100, 1000	Fixed	$C_1 > 0$	$C_2 = 0$	$N(G) = G$	
1, 10, 100, 1000	Estimated	$C_1 = 0$	$C_2 > 0$	Linear	12
1, 10	Estimated	$0 \leq C_1 \leq 1$	$C_2 \geq 0$	$G/(mG + f)$ $\lim_{G \rightarrow \infty} G/(mG + f) = \text{constant}$	13
100, 1000	Estimated	$0 \leq C_1 \leq 1$	$C_2 \geq 0$	Linear	16

Table 3: Scaling behavior in the tree topology

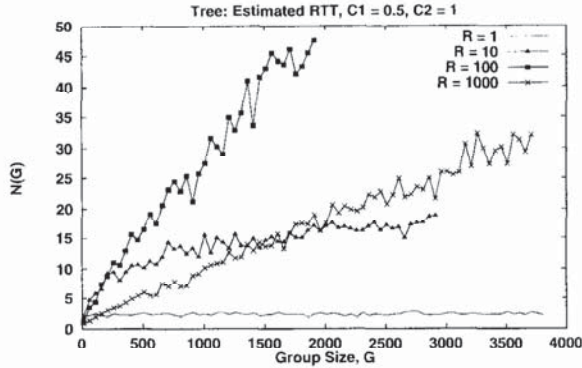


Figure 16:  $N(G)$  in the binary tree with  $\Delta/\delta = 1, 10, 100, 1000$ , RTT estimated,  $C_1 = 0.5$ ,  $C_2 = 1$ .

find that  $C_2$  needs to be “super-linear” in  $D$  to make scaling constant.

## 8 Conclusions

In this paper, we used analysis and simulation to study the scaling behavior of global loss recovery in SRM. The SRM protocol is NACK-based and uses a randomized, timer-based decentralized algorithm to reduce NACK implosion. We use the number of NACKs  $N(G)$  generated in response to a loss, as a metric for scalability. The two protocol parameters,  $C_1$  and  $C_2$ , govern the deterministic and random delays in the firing of a NACK from a receiver. There is a trade-off between low-latency loss recovery and the number of NACKs – in general, making these parameters small leads to lower latency, but usually at the expense of poorer asymptotic scaling. We study  $N(G)$  as a function of group size,  $G$ , for various protocol parameters, on a set of simple, representative topologies — the cone, the linear chain, and the binary tree.

In the cone topology, we find that random backoff is the dominant reason for suppression and scaling is linear. This linear scaling can be reduced by using a distribution that is dependent on the group size. The cone models topologies in which receivers have similar round-trip time estimates to the source. For the linear chain  $N(G)$  is between constant (when  $C_1 > 0, C_2 > 0$ , and RTT estimation is perfect), and logarithmic, when RTT is not estimated. In the tree, scaling is between constant (when  $C_1 > 0, C_2 > 0$ , and RTT estimation is perfect), and linear, when RTT is not estimated. For the linear chain we show that  $C_2 = D^c$  results in constant scaling even when  $C_1 = 0$ , where  $D$  is the one-way delay to the source. Similarly, for the binary tree,  $C_2 = D^2$  results in constant scaling.

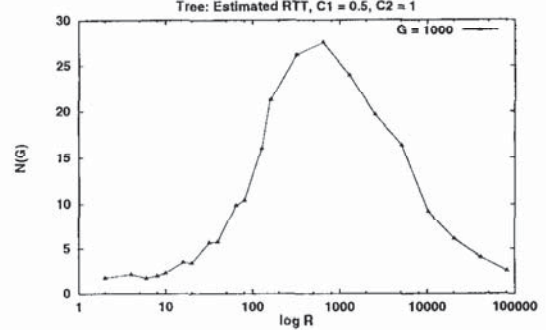


Figure 17: For small values of  $R$ , the round-trip times from the source to the receivers are distinguishable, and deterministic suppression effectively keeps the NACK count low. When  $\Delta/\delta$  increase, randomized suppression is the dominant cause for suppression. The “turning point” value of  $\Delta/\delta$  depends on the topology.

We find that in topologies where deterministic suppression is effective in reducing the number of duplicate NACKs, asymptotic scaling tends to a constant. For topologies in which randomized suppression is mainly responsible for eliminating duplicates, asymptotic scaling is not constant, e.g., in the cone topology and in the binary tree with  $\Delta \gg \delta$ ,  $N(G)$  grows linearly.

In conclusion, we have shown that there is a rich parameter space in the SRM protocol and that the best asymptotic scaling performance is sensitive to the choice of these parameters. We expect our results to be useful in obtaining a better understanding of the reasons for SRM’s scaling properties in different situations, and in aiding the design and analysis of future modifications to SRM and similar protocols that use multicast transmission and suppression.

## 9 Acknowledgements

We would like to thank Hari Balakrishnan and Lee Breslau for extremely useful feedback on this work. At Berkeley, this research was supported by DARPA contract N66001-96-C-8508, by the State of California under the MICRO program, and by NSF Contract CDA 94-01156. At Xerox PARC, this research was supported in part by the Advanced Research Projects Agency, monitored by Fort Huachuca under contracts DABT63-94-C-0073. The views expressed here do not reflect the position or policy of the U.S. government.

## References

- [1] BALLARDIE, T., FRANCIS, P., AND CROWCROFT, J.

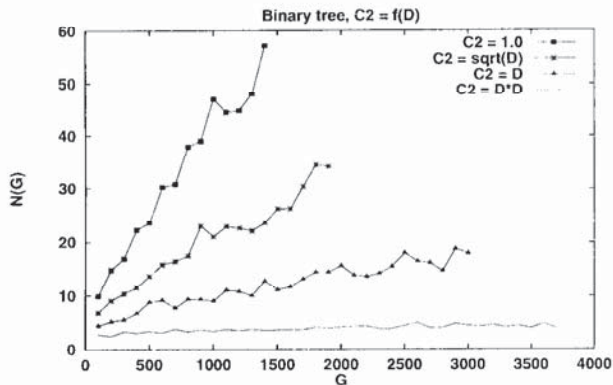


Figure 18: When  $C_2 = D^{0.5}$  in the binary tree,  $N(G)$  has improved scaling.

Core Based Trees (CBT) An Architecture for Scalable Inter-Domain Multicast Routing. In *Proceedings of SIGCOMM '93* (San Francisco, CA, Sept. 1993), ACM, pp. 85–95.

- [2] CHESSEON, G. XTP/protocol engine design. In *Proceedings of the IFIP WG6.1/6.4 Workshop* (Rüschlikon, May 1989).
- [3] CLARK, D. D., AND TENNENHOUSE, D. L. Architectural Considerations for a New Generation of Protocols. In *Proceedings of SIGCOMM '90* (Philadelphia, PA, Sept. 1990), ACM.
- [4] DEERING, S., ESTRIN, D., FARINACCI, D., AND JACOBSON, V. An Architecture for Wide-Area Multicast Routing. In *Proceedings of SIGCOMM '94* (University College London, London, U.K., Sept. 1994), ACM.
- [5] DEERING, S. E. *Multicast Routing in a Datagram Internetwork*. PhD thesis, Stanford University, Dec. 1991.
- [6] FLOYD, S., JACOBSON, V., MCCANNE, S., LIU, C.-G., AND ZHANG, L. A Reliable Multicast Framework for Light-weight Sessions and Application Level Framing. In *Proceedings of SIGCOMM '95* (Boston, MA, Sept. 1995), ACM.
- [7] FREDERICK, R. *Network Video (nv)*. Xerox Palo Alto Research Center. Software on-line <ftp://ftp.parc.xerox.com/net-research>.
- [8] HOLBROOK, H., SINGHAL, S., AND CHERITON, D. Log-Based Receiver-Reliable Multicast for Distributed Interactive Simulation. In *Proceedings of SIGCOMM '95* (Boston, MA, Sept. 1995), ACM.
- [9] JACOBSON, V., AND MCCANNE, S. *LBL Whiteboard*. Lawrence Berkeley Laboratory. Software on-line <ftp://ftp.ee.lbl.gov/conferencing/wb>.
- [10] JACOBSON, V., AND MCCANNE, S. *Visual Audio Tool*. Lawrence Berkeley Laboratory. Software on-line <ftp://ftp.ee.lbl.gov/conferencing/vat>.
- [11] KASERA, S. K., KUROSE, J. F., AND TOWSLEY, D. F. Scalable Reliable Multicast Using Multiple Multicast Groups. In *Proceedings of ACM SIGMETRICS Conference on Measurement & Modeling of Computer Systems* (June 1997).
- [12] LEVINE, B. N., LAVO, D. B., AND GARCIA-LUNA-ACEVES, J. The Case For Reliable Concurrent Multicasting Using Shared Ack Trees. In *Proceedings of ACM Multimedia '96* (Boston, MA, Nov. 1996), ACM.
- [13] LIN, J. C., AND PAUL, S. RMTP: A Reliable Multicast Transport Protocol. In *Proceedings IEEE Infocom '96* (San Francisco, CA, Mar. 1996), pp. 1414–1424.
- [14] LIU, C.-G., ESTRIN, D., SHENKER, S., AND ZHANG, L. Local Recovery in SRM. *Submitted to IEEE Transactions on Networking* (1998).
- [15] LIU, C.-G., ESTRIN, D., SHENKER, S., AND ZHANG, L. Recovery Timer Adaptation in SRM. *Submitted to IEEE Transactions on Networking* (1998).
- [16] MCCANNE, S., AND FLOYD, S. *The LBNL Network Simulator*. University of California, Berkeley. <http://www-mash.cs.berkeley.edu/ns/>.
- [17] MCCANNE, S., AND JACOBSON, V. *vic: video conference*. Lawrence Berkeley Laboratory and University of California, Berkeley. Software on-line <ftp://ftp.ee.lbl.gov/conferencing/vic>.
- [18] NONNENMACHER, J., AND BIRSACK, E. W. Optimal Multicast Feedback. *IEEE Infocom* (1998).
- [19] PINGALI, D., TOWSLEY, D., AND KUROSE, J. F. A comparison of sender-initiated and receiver-initiated reliable multicast protocols. In *Proceedings of SIGMETRICS '94* (Santa Clara, CA, May 1994).
- [20] RAMAN, S., AND MCCANNE, S. Generalized Data Naming and Scalable State Announcements for Reliable Multicast. Tech. rep., University of California, Berkeley, CA, June 1997.
- [21] STEVENS, W. R. *TCP/IP Illustrated, Volume 1 – The Protocols*, first ed. Addison-Wesley, Dec. 1994.
- [22] TURLETTI, T. *INRIA Video Conferencing System (ivs)*. Institut National de Recherche en Informatique et en Automatique. Software on-line <http://www.inria.fr/rodeo/ivs.html>.
- [23] YAJNIK, M., KUROSE, J., AND TOWSLEY, D. Packet Loss Correlation in the Mbone Multicast Network. *IEEE Global Internet Conference* (1996).
- [24] YAVATKAR, R., GRIFFIOEN, J., AND SUDAN, M. A Reliable Dissemination Protocol for Interactive Collaborative Applications. In *Proceedings of ACM Multimedia '95* (San Francisco, CA, Nov. 1995), ACM.