Scalability of Multicast Communication over Wide-Area Networks

Donald Yeung
Laboratory for Computer Science
Cambridge, MA 02139

April 24, 1996

Abstract

A multitude of interesting applications require multicast over wide-area networks. As these applications grow in popularity, issues concerning scalability of the underlying protocols that support both unreliable and reliable multicast become important. This paper makes several intellectual contributions related to scalable multicast protocols. First, we present a model that captures the essential network and application parameters. The model enables the direct comparison of different unreliable multicast routing algorithms under a single consistent framework. Second, we use the model to conduct an in-depth analysis of four existing unreliable multicast routing algorithms. Among our findings is the somewhat surprising result that link-state multicast protocols are scalable under certain conditions because data message cost dominates control message cost, even when groups are sparse, and group membership is dynamic. Third, the paper presents an analysis of the cost of reliable multicast in two competing protocols. We identify NACK suppression as a problem for reliable multicast. Finally, we propose a novel solution for the suppression of multicasted NACKs that is implemented at the network level, and suggest directions for future research in this area.

1 Introduction

Multicast communication over wide-area networks is proving to be an important network service. A multitude of interesting applications demand wide-area multicast, including internet TV/radio, video/audio conferencing, and distributed interactive simulation, just to name a few. While there is already very high interest in the MBone [9], the Internet's version of multicast, this is only an early indication of the future popularity for applications requiring multicast.

As the demand for multicast in networks increases, the question of scalability needs to be addressed. Protocols that support both unreliable and reliable multicast over wide areas have the potential for consuming enormous amounts of bandwidth. As network size, group size, and application bandwidth scale, it is easy for a multicast protocol to bring a network to its knees. Therefore, it is important to understand how scaling various network and application parameters impacts different multicast protocols.

This paper explores scalability issues in both unreliable and reliable multicast, relying heavily on analysis to provide intuition. The paper makes several contributions. First, a model is introduced for reasoning about unreliable multicast protocols. The model considers both network and application parameters, and enables a direct comparison of different protocols under a single consistent framework. Second, we use the model to study the scalability of four proposed unreliable multicast protocols. A somewhat surprising result is that link-state protocols are scalable under certain conditions because data message cost dominates control message cost, even when groups are sparse, and group membership is dynamic. Third, the paper provides an in-depth comparison of two existing reliable multicast protocols, and identifies NACK suppression as a problem for reliable multicast. Finally, we propose a novel solution for the suppression of multicasted NACKs that is implemented at the network level.

Section 2 discusses unreliable multicast, and Section 3 discusses reliable multicast. We conclude the paper in Section 4.

2 Unreliable Multicast

This section discusses unreliable multicast communication over wide-area networks. Section 2.1 presents our assumptions, and Section 2.2 gives some background on existing routing techniques. Section 2.3 presents a model to reason about the cost of unreliable multicast, and Section 2.4 uses the model to study the scalability of the techniques described in Section 2.2. Section 2.5 then discusses other scalability issues not captured by our model, and Section 2.6 summarizes the findings in our analysis.

2.1 Assumptions

We are concerned with unreliable multicast routing across point-to-point links at the internet level. We assume that protocols exist for gateways to acquire membership information from hosts in their subdomains (for instance, the protocol in [6]), and that there are separate protocols for forwarding multicast packets from gateways to hosts at the local-area level. Also, we assume that logical addressing is used so that senders need not worry about the existence of group members.

2.2 Routing Algorithms for Unreliable Multicast

This section describes, briefly, four algorithms for multicast routing: Reverse Path Multicast (RPM), Link-State Multicast, Core-Based Trees (CBTs), and Protocol Independent Multicast (PIM).

2.2.1 RPM

Reverse Path Multicast (RPM) relies on distance vector routing to form multicast distribution trees. In a distance vector routing algorithm, the routing tables in each router consists of distance vectors. Each routing table entry contains a destination, its distance from the router, and the outgoing link used to reach the next hop along the shortest path to the destination [13]. A unicast packet is routed by finding the distance vector corresponding to its destination, and then forwarding the packet through the outgoing link specified in the distance vector. For multicast routing, source-rooted shortest-path trees (SPTs) can be formed from the same routing tables used for unicast routing [5]. Given a source, any router can determine its parent and children nodes in the SPT rooted at the source in the following way. The parent node is simply the router found in the next hop towards the source, and the children are the routers who's next hop to the same source is the given router. A tree formed in such a way is known as the *shortest-path reverse tree*. RPM uses this tree to copy and forward multicast packets.

The problem with reverse-path trees is that routers have no knowledge of which children links lead to group members. This can result in many wastefully forwarded messages. In RPM, gateways at the leaves keep track of group membership information for the local networks that are connected to the gateway. The gateway sends prune messages up the reverse-path tree if they receive multicast messages for which they have no members [8]. However, this does not solve the problem of wasteful messages. In order for group members to "learn" about the existence of new sources, any pruned branch must periodically time out so that a multicast from a new source can make its way down to the leaves and "announce" itself [27]. This is necessary because in source-rooted trees, there is a different tree for every source.

2.2.2 Link-State Multicasting

One problem with distance vector routing is that it is unstable in large networks during network topology reconfiguration [17]. This has motivated the proposal of a new method of routing, called link-state routing. In link-state routing, each time a link goes down or comes up, a broadcast to the entire network occurs notifying all routers. Therefore, all routers have global knowledge about the network topology.

Adding multicast to link-state routing is easy: just make group membership part of the link state that gets broadcasted. This way, routers have global knowledge about group membership as well. Each router has all the information necessary to compute the SPT given any source and any set of group members. MOSPF [19] is such a link-state multicast protocol.

Because every router in a link-state protocol has global group membership knowledge, wasteful copying and forwarding of multicast packets down links that do not lead to group members can be avoided. The disadvantage of link-state protocols is that the computation of SPTs can be expensive, especially since there are $S \times N$ trees, where N is the number of groups, and S is the number of sources per group. Instead of computing all possible trees, the trees are typically computed on demand and cached. This inflicts a longer latency to the first multicast packet that traverses a new tree. Also, link-state protocols can be expensive because each time a group member joins or leaves the group, a broadcast to the entire network occurs.

2.2.3 CBTs

A problem with shortest-path trees is that there is a different distribution tree for every group, and every source in the group. Therefore, the router state needed to track these trees (in MOSPF, for instance) grows as $S \times N$. To address the scalability problem this presents, Core Based Trees (CBT) has been proposed [2]. The idea is that each group, regardless of the number of sources, uses a single shared tree for multicast distribution, rooted at a fixed and agreed upon router, called the "core." During a multicast, a source sends a multicast packet to the core. This packet is routed without any copying. When a multicast reaches the core, it is delivered to all group members through a single shared tree rooted at the core that spans all group members.

CBTs have the advantage that sources and receivers can easily "find" each other because there is a fixed core. Sources' multicast packets are automatically forwarded to the core, and receivers that want to subscribe simply send join messages towards the core. Also, CBTs provide very good "information hiding." That is, only routers that are a part of a distribution tree are burdened with messages related to that tree. This is not the case with RPM or MOSPF. The disadvantage of CBTs is that the average path length from source to receivers is typically longer than in an SPT. Also, if there are many sources, all their packets are funneled through the core, potentially causing congestion. Known as "bandwidth concentration," this effect is less severe in SPTs because each source has its own distribution tree.

2.2.4 PIM

Protocol Independent Multicast (PIM) [7] combines shortest-path trees and shared trees. PIM recognizes that SPTs have good path delay and bandwidth concentration characteristics. On the other hand, shared trees make it easy for sources and receivers to "find" each other, and have good information hiding characteristics. In PIM, there is a fixed router that initiates multicast between a source and a group of receivers. This router is called the "rendezvous point" (RP), and is analogous to the core in CBTs. Receivers join a group by sending join messages to the RP. Sources always multicast to the RP, so new receivers will begin receiving multicast packets from all existing sources via a shared tree rooted at the RP. When a receiver starts getting multicast packets, it can decide to switch over to a shortest-path tree by sending a join towards the source. When the source receives such a join request, it starts sending multicast packets through the shortest-path tree. The shared tree originally used to reach the receiver is pruned once the receiver has switched to an SPT.

PIM is largely successful at achieving the advantages of both tree types. However, it still has the $S \times N$ routing state scaling in the routers, and there is the problem of choosing RPs. Also, the RPs can be a point of failure. A solution is to replicate RPs, but this presents more complexity to the protocol.

2.3 A Framework for Reasoning about Unreliable Multicast

Existing multicast routing algorithms are very different. Each makes several tradeoffs in how distribution trees are maintained, and how multicast packets are forwarded. While any one tradeoff is simple to understand, the interplay of several tradeoffs can be very unintuitive. No framework exists to enable an understanding of the sum total of all these effects at once.

We present an analytic model that captures the essential parameters, both in the network and in the application, that affect the performance of unreliable multicast routing algorithms. The model allows a direct comparison under a consistent framework of the algorithms discussed in Section 2.2. We use the model to focus on scalability. The model presented in Section 2.3.1 only deals with unreliable multicast; analysis of reliable multicast is discussed later in the paper.

2.3.1 The Model

Our model measures the **communication cost** of a multicast routing algorithm in terms of the total bandwidth consumed in the network by the routing algorithm. Our notion of cost is similar to the work in [29]; however, our model is novel because it considers the cost of maintaining multicast distribution trees, as well as the cost of transferring data. We express cost as:

$$msgCost = C_{cntrl} f_{cntrl} + (C_{tree} + C_{nm}) BW_{send} \tag{1}$$

The first term in Equation 1 expresses the control messaging cost, while the second term expresses the data messaging cost. Control and data messaging costs are defined as follows:

Control Message Cost. Control messages are exchanged between network routers to disseminate group membership information. C_{cntrl} is the total number of point-to-point messages communicated each time a member joins or leaves the group. f_{cntrl} is the frequency at which members join or leave. [1]

[1] We assume that a group reaches a steady-state in which the number of joins and departures balance. Therefore, the group

Data Message Cost. Data messages consist of all the messages that carry data from a multicast application. For every multicast message injected by an application, there is some number of point-to-point messages that traverse the network due to copying and forwarding at each intermediate router between the source and all the receivers. This number is $C_{tree} + C_{nm}$. A point-to-point message is counted in the C_{tree} component if it leads to at least 1 group member downstream; otherwise, it is counted in the C_{nm} (non-member) component. BW_{send} is the multicast rate of the application.

In our model, total bandwidth cost is measured as messages per second. As a simplification, we assume that all messages have the same size. Applications that multicast large data messages can be modeled by a higher BW_{send} parameter. Also, the model assumes an ideal network with infinite bandwidth so that no queuing or blocking occurs at any of the router nodes.

f_{cntrl} and BW_{send} are application parameters, while C_{cntrl}, C_{tree}, and C_{nm} are different for each routing algorithm. We describe the derivation for the MOSPF protocol here, and provide the derivation for RPM, CBT, and PIM in Appendix A.

The number of control messages for MOSPF is given by:

$$C_{cntrl}^{MOSPF} = N(\alpha - 1) \tag{2}$$

Each time a member joins or leaves the group, MOSPF broadcasts this information by flooding. At each router, the flooding algorithm sends a message out every outgoing link except for the direction from which the message came in. Therefore, there are $\alpha - 1$ messages sent at each router, where α is the average node degree. Since there are N routers, we have Equation 2.

The cost of distributing data messages that lead to group members in MOSPF is:

$$C_{tree}^{MOSPF} = C_{SPT}(N, G) \tag{3}$$

Given a network, N, and a group, G, $C_{SPT}(N, G)$ is the number of point-to-point messages needed to multicast to G using a shortest-path tree rooted at the sender, averaged over all possible senders in G. There is no simple expression for $C_{SPT}(N, G)$ as it is highly dependent on the particular topology of the network, the size of the group, and the location of group members. In our study, we compute $C_{SPT}(N, G)$ in the following manner. Given a network topology and a group size, we randomly place the group members, choose a sender, and compute the cost of the shortest-path tree. We do this for all possible choices of senders for a given group placement, and for several random placements of the group. $C_{SPT}(N, G)$ is the average over all such shortest-path tree costs.

Finally, because MOSPF has global knowledge about group membership at each router, it can avoid sending messages down links that do not lead to group members. Therefore, there is no C_{nm} component in MOSPF:

$$C_{nm}^{MOSPF} = 0 \tag{4}$$

2.4 Analyzing Scalability Using the Model

In this section, we use the model presented in Section 2.3.1 to investigate the impact on cost to each of the routing algorithms when the following parameters are scaled:

size remains the same, but each join or departure still incurs control message overhead.

Number of Nodes	200
Average Node Degree	3.79
Diameter	11
$C_{SPT}(N_{200}, G_{10})$	26.63, $\sigma = 2.70$
$C_{SPT}(N_{200}, G_{160})$	145.49, $\sigma = 3.97$
$C_{CBT}(N_{200}, G_{10})$	24.52, $\sigma = 2.46$
$C_{CBT}(N_{200}, G_{160})$	140.60, $\sigma = 4.24$

Table 1: Statistics on the 200-node random graph. For all distribution tree cost calculations, the average is given along with the standard deviation, σ.

- Application bandwidth, by increasing BW_{send}.
- Group sparseness, by decreasing $\frac{|G|}{|N|}$.
- Join/leave frequency, by increasing f_{cntrl}.
- Flood duty factor, by increasing $flood_{duty}$. [2]
- Network size, by increasing $| N |$.

We begin our analysis by considering a medium-sized network consisting of 200 routers. A random network of 200 nodes was generated using the method described in [28], and the average cost for shortest-path trees and core-based trees was computed. We assume each router is attached to a local-area network with multiple hosts that can subscribe to a group. If a router has at least 1 local host subscription, we count the router as a single group member at the internet level. Table 1 shows some statistics describing this graph. Originally, we conducted our analysis on a smaller network, the 1976 Arpanet, which consists of 59 nodes. We found that the scaling issues we study do not matter in such a small network, and omit the results in the interest of space.

Figure 1 shows how the routing algorithms scale with application bandwidth on networks with dense groups. We plot the messaging cost, Equation 1, as a function of application bandwidth for each of the routing algorithms. The group size is 160 nodes with members, and the join/leave rate is low, 1 join/leave every 10 seconds. Also, the flood duty for RPM has been set to 0.01, meaning that RPM does broadcast 1% of the time.

The only visible difference (and barely at that) is the difference in slopes between the curves for MOSPF, RPM, and PIM as compared against the curve for CBT. This is because MOSPF, RPM, and PIM all use shortest-path trees which are slightly more costly than the optimal-delay core-based tree used by CBT, as shown in Table 1. Therefore, in our model, C_{tree}^{MOSPF}, C_{tree}^{RPM}, and C_{tree}^{PIM} are slightly higher than C_{tree}^{CBT}. This difference is extremely small and causes less than $\frac{1}{10}$ of one percent difference in cost between all the algorithms.

There are two surprises in Figure 1. First, the broadcast of group membership in MOSPF each time a host joins or leaves the group has no measurable effect. For our 200 node network, each join/leave costs MOSPF 558 point-to-point messages. This does not significantly impact the overall cost of MOSPF because the join/leave frequency is low, 1 every 10 seconds, and thus joins/leaves generate very few messages relative to the number of data messages generated by the application. Second, the periodic flooding that occurs

[2] $flood_{duty}$ is the fraction of time RPM spends broadcasting instead of multicasting. For an explanation, see appendix Section A.1.

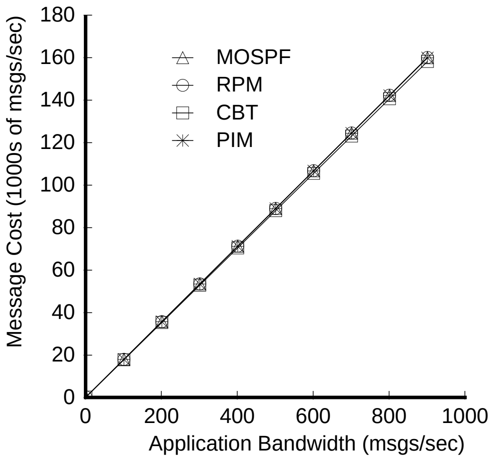

Figure 1: Performance of unreliable multicast on a 200-node graph. Group size = 160, f_{cntrl} = 0.1, $flood_{duty}$ = 0.01.

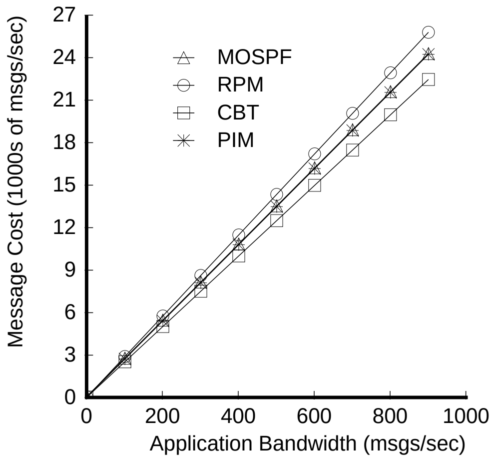

Figure 2: Performance of unreliable multicast on a 200-node graph. Group size = 10, f_{cntrl} = 0.1, $flood_{duty}$ = 0.01.

in RPM has no measurable effect either. In a dense group, the cost of reverse-path broadcast is not much higher than the cost of a multicast since most of the network nodes are already involved in the multicast. Furthermore, the flooding duty factor, 1%, is small, so the broadcast happens infrequently.

Figure 1 shows that the protocols have comparable scaling when the group is dense, the join/leave frequency is low, and the flood duty factor is low. In the remainder of our study, we will relax these assumptions. We start by looking at sparse groups. In Figure 2, all things stay as they were in Figure 1 except that group size is reduced from 160 nodes to 10 nodes. The effect is that the slopes of all the graphs are spread a bit more. MOSPF and PIM have a higher slope than CBT because of a greater cost difference between SPT-based trees as compared against CBT-based trees under low density (see Table 1). RPM is also impacted by the higher relative cost of shortest-path trees, but in addition, RPM pays a higher cost for periodically flooding. When the density of groups is low, there is a significant cost difference between a broadcast and a multicast, which was not the case for dense groups.

Although there is a larger difference between the protocols as compared to Figure 1, the difference is still small, only about 17% separating the best and worst costs. While many claim that MOSPF and RPM are not feasible in sparse groups [7, 2], our analysis shows that they are competitive when join/leave frequency and flood duty are small.

The cost of MOSPF and RPM is controlled by keeping join/leave frequency and flood duty small. What happens when these assumptions are relaxed? In Figure 3 the join/leave frequency has been increased to 10 joins/leaves per second, and the flooding duty factor to 10%. First, we notice that MOSPF has a large fixed overhead. At 10 joins/leaves per second, MOSPF is paying 5580 control messages per second just to keep track of the membership information. At small application bandwidths, this cost dominates, but as application bandwidth increases (and the join/leave frequency remains constant), the impact becomes less and less significant, down to about 35% at BW_{send} = 1000 msgs/sec. We have also noticed that a high join/leave frequency does not impact MOSPF in dense groups, but we omit this analysis in the interest of space.

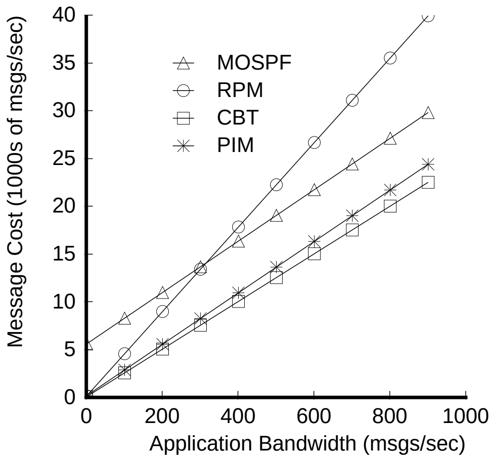

Figure 3: Performance of unreliable multicast on a 200-node graph. Group size = 10, f_{cntrl} = 10, $flood_{duty}$ = 0.1.

The cost of a high flood duty factor to RPM is more severe. When the flooding duty factor is 10%, the cost of flooding is almost double the cost of multicasting. RPM maintains a steady 80% higher cost as compared against CBT and PIM. An interesting point is that there is a cross-over between the costs of MOSPF and RPM where the join/leave frequency impacts MOSPF the same amount as the flooding cost impacts RPM.

Thus far, our analysis has shown that CBT and PIM scale well in both dense and sparse groups, while MOSPF and RPM remain competitive only under certain conditions. We now focus on MOSPF and RPM, and see under what conditions they remain competitive on a large network with 10,000 routers. Unfortunately, the computation of $C_{SPT}(N, G)$ and $C_{CBT}(N, G)$ become intractable when the network size and group size are large. Since we are interested in sparse groups, as this stresses the protocols the most, we consider $\mid G \mid \ll 10,000$ and make the assumption that $C_{SPT}(N_{10,000}, G) \approx C_{SPT}(N_{200}, G)$ and $C_{CBT}(N_{10,000}, G) \approx C_{CBT}(N_{200}, G)$ for $\mid G \mid \leq 200$. [3]

Figure 4 shows the behavior of MOSPF in a 10,000 node network with a group size of 160. Normalized message cost is plotted against group join/leave frequency for three application bandwidths on a semi-log scale. The normalization factor is the message cost of the CBT protocol at the same join/leave frequency, and the same application bandwidth. Figure 4 shows two interesting points. First, the message cost does not blow up when the application bandwidth is high. When application bandwidth is high, join and leave events are insignificant, even when they're frequent, because data message cost dominates control message cost. Second, when application bandwidth is moderate, the only way to keep MOSPF from blowing up in sparse groups is to keep the join/leave rate very low. For low application bandwidths, MOSPF is unscalable in large networks.

Figure 5 shows the behavior of RPM under the same network and group size. We plot normalized message cost versus the flood duty factor using the same normalization method used for MOSPF. The shape of this curve is independent of application bandwidth since when RPM uses broadcasting, the bandwidth consumed is proportional to the application bandwidth. Figure 5 shows that message cost for

[3]Notice this assumption is pessimistic towards MOSPF and RPM. In effect, the assumption makes the group membership look even more sparse because for the same number of group members, the distribution tree in the 10,000 node graph will in actuality be larger than for the 200 node graph.

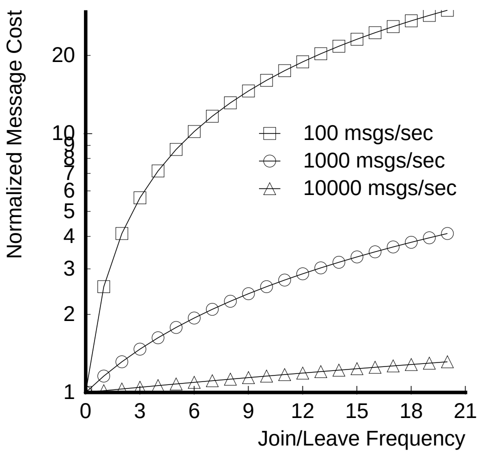

Figure 4: Normalized message cost versus group join/leave frequency for several application bandwidths in MOSPF. Network size = 10,000 nodes, group size = 160.

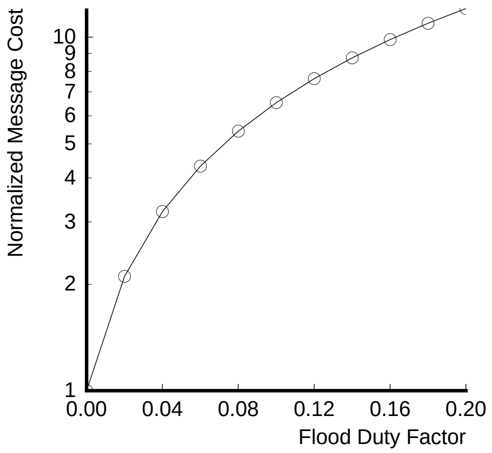

Figure 5: Normalized message cost versus flood duty factor for in RPM. Network size = 10,000 nodes, group size = 160.

RPM can be managed in large networks only by keeping the flood duty factor very small.

2.5 Other Scalability Issues

There are two important scalability issues not considered by our model. The first is the scalability of router table state. RPM and CBT impose the least space requirements on router tables. In RPM, the shortest-path tree is computed by computing the shortest reverse-path tree. The shortest reverse-path tree can be derived from a standard routing table used by distance-vector routing [8], thus, there is no added cost for multicast. RPM does incur routing state costs for tracking pruned branches. In CBT, each router records the children it must copy and forward a multicast packet to. Since there is only one tree per group in CBT, the router table state grows in proportion to the number of groups. PIM makes somewhat higher demands on router table state. The problem is that shortest-path trees are different depending on where the sender is located. Therefore, the amount of state required to track all trees is proportional to the product of the number of groups, and the number of senders in each group. Finally, MOSPF has the same routing table scaling as PIM. In addition, MOSPF must record the entire group membership information for each group at every router.

Our model assumes that there is infinite bandwidth on every link in the network, and measures costs in terms of total bandwidth consumed. When bandwidth is limited, or when different links offer very different bandwidths, it is important to talk about maximum bandwidth on a per-link basis. In general, core-based trees have worse maximum bandwidth load because of the bandwidth concentration effect described in Section 2.2.3. If there are multiple senders, the accumulated bandwidth at the root of the shared tree can be high. Our model does not penalize CBTs for bandwidth concentration. Furthermore, protocols like MOSPF that send control messages to every corner of the network can adversely impact parts of the network with "weak links." Our model only considers the total bandwidth consumed by such control messages. Our model can be changed to account for bandwidth on a per-link basis. For instance, the cost of a message on a point-to-point link can be weighted by some function of the number of other messages sharing the link, and by the fraction of bandwidth that can be reasonably supported by the link that has been consumed.

2.6 Summary of Scalability Study

In this section, we have developed an analytic model that allows us to reason about the cost of different unreliable multicast routing algorithms. Through the use of our model, we have learned the following about the scalability of four existing multicast routing algorithms:

- The difference in cost between SPTs and CBTs is small, although more significant for sparse groups. In any case, this discrepancy is a small issue in regards to scalability.
- MOSPF must be careful about join/leave frequency. However, the cost of broadcasting every join or leave may be insignificant compared to data message cost. Data message cost will dominate control message cost even if joins and leaves are frequent if application bandwidth is high, or in the case that application bandwidth is moderate, if groups are dense. In situations where groups are sparse and application bandwidth is low, MOSPF is competitive only if join/leave frequency is low. MOSPF does have adverse effects in corners of the network with "weak links." Our model does not capture this effect, but can be extended to account for different per-link bandwidths.
- RPM must be careful about flooding duty factor. In dense groups, this is not an issue because a broadcast is similar in cost to a multicast. In sparse groups, RPM is competitive only if the flood duty factor is low.
- CBT scales well under all our analyses. Since our model considers only total bandwidth, the result for CBT must be qualified by considering bandwidth concentration. If the number of senders is large, the core may become a bottleneck making CBT unscalable. An extension to our model could provide this analysis.
- PIM scales well under all our analyses. PIM gets good marks because it scales well and allows the use of shortest-path trees for multicast.

3 Reliable Multicast

This section discusses reliable multicast communication. Section 3.1 briefly describes our assumptions for reliable multicast. Section 3.2 then describes two competing approaches, and Section 3.3 compares them. Finally, Section 3.4 proposes network-level mechanisms to help receiver-reliable schemes address the NACK suppression problem.

3.1 Assumptions

There is a large body of work on reliable broadcast and multicast from the distributed systems community [15, 3, 31]. These systems have traditionally operated over local-area networks, often with support for broadcast in the network, and guarantee a high degree of reliability. We assume that for wide-area multicast, the network should only provide a minimum degree of reliability: reliable delivery of packets, in any order, without a great concern for faults. The construction of stricter reliability guarantees is left up to the application.

We also assume that the correct model of reliability for multicast is receiver reliability, as opposed to source reliability. While both techniques have been used successfully in unicast communication [4, 21], receiver reliability is better suited for multicast communication. The main reason is that receiver reliability does not issue positive ACKs for every packet sent, but instead, only negatively ACKs those which have been detected to have been lost. This avoids the well-known "ACK implosion" problem. (For more information, see [20]).

3.2 Receiver-Reliable Multicast Techniques

3.2.1 LBRM

Holbrook et. al. propose a receiver-reliable scheme known as Log-Based Receiver-Reliable Multicast (LBRM) [14]. In log-based receiver reliability, a source multicasts to the group, and uses periodic heartbeats so that receivers can distinguish between lost packets and an idle sender. At the same time, it reliably unicasts (using source-reliable unicast) the same packets to a centralized logging server. Once the source has reliably sent a packet to the logging server, it can discard it from its buffer. The logging server has the resources to maintain a log of all packets sent by the source. A receiver that detects lost packets, either by noticing a missing sequence number or by not receiving any packet for a heartbeat interval, sends a NACK for the missing packet(s) to the logging server which is responsible for retransmitting any lost packets. Because wide-spread loss of a multicast can happen if a packet is lost near the root of the distribution tree, there is a potential for "NACK implosion" at the central logging server. LBRM proposes two optimizations that address the NACK implosion problem.

The first optimization replicates the central logging server, known as the "primary server," and places a "secondary server" at each domain site. The secondary servers receive multicasts (possibly unreliably) through the normal distribution tree, and logs them. When a receiver detects a lost packet, it requests a retransmit from the secondary server in its local domain instead of going to the primary server. The local secondary server tries to satisfy the retransmit. If it can't because it didn't reliably receive the packet itself, it sends a single NACK to the primary server regardless of the number of NACKs it gets from the domain. The secondary server provides two benefits. First, it prunes NACKs to the primary server since at most one NACK is sent from each domain. Second, it reduces the average error recovery latency seen by receivers since the round trip time (RTT) to a local secondary server is less than to the primary server.

The second optimization tries to select an appropriate retransmit strategy. The idea is that if the loss of a packet is wide spread, it is more efficient to retransmit via multicast; however, if a packet is only lost by a few receivers, retransmits should be unicasted on demand. In the case that multicast is chosen, if it is done early enough, the retransmit message can suppress NACK generation at many of the receivers, thus preventing NACK implosion. Holbrook proposes to choose the retransmit strategy based on statistical acknowledgments. A small group of secondary servers are selected at random, known as "Designated Ackers." After a multicast, each Designated Acker immediately ACKs the multicast packet. The number of ACKs that are received is used by the sender as a statistical indication of how widely the packet was reliably received. If the number of ACKs is below a certain threshold, the sender retransmits immediately using multicast. Otherwise, the sender waits for NACKs to arrive, and handles them using a unicast retransmit strategy.

3.2.2 SRM

Scalable Reliable Multicast (SRM) is proposed in [10] as a solution to reliable multicast for the distributed whiteboard application. The contribution of SRM is a novel NACK-retransmit strategy. In SRM, when a receiver detects a lost packet, the NACK is multicasted to the entire group. Subsequently, any group member that has reliably received the NACKed packet can reply to the NACK, not only the sender. Furthermore, the retransmit itself is multicasted to the entire group as well thus potentially satisfying multiple NACKs at once. This approach distributes the retransmit responsibility amongst all the group members; therefore, the number of hosts that can reply to a NACK naturally grows with group size.

By using multicast for both the NACK and the retransmit, SRM has potentially made the NACK implosion problem worse in that everyone in the group can be flooded with NACKs, not just the sender. There's a similar problem with retransmits since more than one host can decide to reply to a NACK.

SRM solves the NACK suppression problem in the following way. Because NACKs use multicasting instead of unicasting, it is possible for a receiver that has lost a packet to suppress its NACK if it "hears" a NACK for the same packet from another receiver. An identical technique can be used to suppress redundant retransmits. The problem of a flood of multicasts can still happen if NACKs are issued simultaneously by multiple receivers. SRM avoids this by desynchronizing NACK requests from different receivers. When a receiver wishes to issue a NACK, it sets a timer and issues the NACK only if the timer goes off. Meanwhile, if it "hears" a NACK for the same packet, it resets the timer with some backoff. An attempt is made to skew different timers by using a randomization technique. The hope is for only one timer to expire, and for the NACK subsequently issued to suppress all other NACKs. The effectiveness of similar desynchronization techniques are explored in [20].

3.2.3 Other Schemes

Other receiver-reliable schemes include Negative Acknowledgment with Periodic Polling (NAPP) [23], and the Multicast Transport Protocol (MTP) [1]. We only mention them briefly here because they were not designed to scale to the extent that LBRM or SRM scale. NAPP multicasts NACK requests to allow multiple receivers to track the status of all receivers. In addition, NAPP requires costive ACKs from all receivers periodically (with a low frequency) so that the source can discard buffered packets that have been received by all members. MTP also addresses source buffering issues simply by allowing the source to discard packets after a certain interval has passed.

3.3 Analyzing LBRM and SRM

In this section, we take a closer look at the two schemes, LBRM and SRM, and provide insight into the two approaches. Again, we focus on messaging cost in the network. We assume that some cost is incurred for unreliable multicast, as described in Section 2.3. In this analysis, we want to know the *additional* cost needed to add reliability.

To aid us in our analysis, we use a simple model for the reliable delivery of messages. We assume that each member of a multicast group receives a message reliably with probability $(1-p)$; therefore, p is the probability of loss at any given receiver. We assume that the reliable reception of a multicast message at different receivers are independent events. This is a simplification that only models loss at the leaves of the multicast distribution tree. In actuality, when a multicast message is lost, the entire subtree downstream from the point of loss does not receive the message. However, studies on the MBone [32] have shown that most messages are lost in tail circuits leading to leaf gateways, not in the backbone. Let F be the the number of receivers that do not reliably receive a given multicast. F is a random variable with a binomial distribution:

$$P\left[F=n\right]=\left(\begin{array}{c}\mid G\mid\\ n\end{array}\right)(1-p)^{|G|-n}\,p^{n} \tag{5}$$

We make the following simplifying assumptions. First, we only consider losses at the internet level, and treat each gateway as if it were a single receiver. Second, we assume that the NACK and retransmit suppression techniques used in SRM are perfect. That is, when a multicast is not reliably received by any group member, there is exactly one NACK and one retransmit of the data. Third, we assume that the statistical acknowledgment technique used in LBRM to guess the number of lost messages is perfect. That is, the primary server knows exactly how many receivers did not receive the multicast reliably. Moreover, we will assume that the messaging needed to implement the statistical acknowledgment technique is negligible. Finally, we assume that all multicasts in both schemes use shortest-path trees.

We now write the expected message cost for reliability under the two schemes. For SRM, two multicasts are transmitted (one for the NACK, and one for the retransmit) if *at least* one receiver does not reliably receive the multicast, and zero messages are sent if all receivers reliably receive the multicast:

$$\begin{aligned}
E\left[msgOvhd_{SRM}\right] &= 2C_{SPT}(N,G)P\left[F\geq 1\right]+0P\left[F=0\right]\\
&= 2C_{SPT}(N,G)\left(1-P\left[F=0\right]\right)\\
&= 2C_{SPT}(N,G)\left(1-\left(\begin{array}{c}\mid G\mid\\ 0\end{array}\right)(1-p)^{|G|}\,p^{0}\right)\\
&= 2C_{SPT}(N,G)\left(1-(1-p)^{|G|}\right) \qquad (6)
\end{aligned}$$

For LBRM, there is a threshold, T_{loss}, that defines what the protocol will do. If the number of receivers that did not reliably receive the multicast is less than T_{loss}, LBRM will service the NACKs one by one, using unicast communication. In this case, there are two unicast messages sent per NACKing receiver (one for the NACK, and one for the retransmit). We pessimistically assume that a unicast between a receiver and the primary server traverses $\frac{1}{2}diam(N)$ hops. If, however, the threshold is met or exceeded, a multicast will be issued to suppress the generation of multiple NACKs. We have for LBRM:

$$E\left[msgOvhd_{LBRM}\right]=\sum_{n=1}^{T_{loss}-1}2n\frac{1}{2}diam(N)P\left[F=n\right]+\sum_{n=T_{loss}}^{|G|}C_{SPT}(N,G)P\left[F=n\right]$$

$$=diam(N)\sum_{n=1}^{T_{loss}-1}n\left(\begin{array}{c}\mid G\mid\\ n\end{array}\right)(1-p)^{|G|-n}\,p^{n}+C_{SPT}(N,G)\sum_{n=T_{loss}}^{|G|}\left(\begin{array}{c}\mid G\mid\\ n\end{array}\right)(1-p)^{|G|-n}\,p^{n} \tag{7}$$

In Equation 7, given a network, a group size, $\mid G\mid$, and a failure probability, p, it is possible to solve for T_{loss} such that the message cost is minimized.

Figure 6 plots the messaging cost under SRM and LBRM for two different per-receiver loss probabilities, 0.01 and 0.1, over a range of group sizes. The network topology we use is the same 200 node network

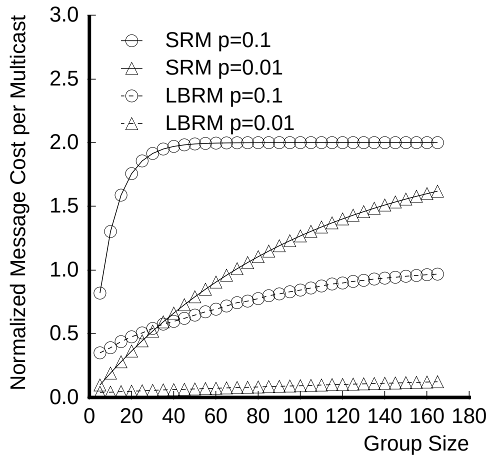

Figure 6: Normalized messaging cost for reliability versus group size under SRM and LBRM on a 200 node network. Per-receiver loss probabilities of 0.01 and 0.1 are plotted for each.

used in Section 2.3 to analyze unreliable multicast routing. For LBRM, we solve for the value of T_{loss} that minimizes message cost given unique values of $\mid G \mid$ and p. Each message cost value in Figure 6 has been normalized by $C_{SPT}(N, G)$. Thus, the normalization value varies along the x-axis, but the same value is used for all curves at a given point on the x-axis.

Figure 6 shows three interesting features. First, under the same loss probabilities, LBRM always has a lower message cost than SRM. Second, the message cost for each scheme is asymptotic. The SRM scheme approaches the value 2, while the LBRM scheme approaches the value 1. This is expected because in the worst case, SRM sends 2 multicasts and LBRM sends 1 multicast. Finally, we find that the cost for LBRM degrades much more gradually than for SRM. This is particularly visible when the loss probability is high: SRM reaches its asymptotic value almost immediately, while in LBRM, the asymptote is reached only at large group sizes. This is because in SRM, as long as there is one loss, the maximum overhead is incurred. However, in LBRM, the protocol doesn't switch over to multicasting unless it is profitable. When the number of losses is small, unicasting is used.

While Figure 6 provides intuition, it is an idealized picture because of the assumptions made. In actuality, the comparison is even worse for SRM. First, desynchronization between multiple NACK requests and retransmits is imperfect, so the asymptote for SRM will be greater than 2, and depends on the group size and network topology. Second, by treating each gateway as a single receiver, we have removed the benefit of cheaper recovery through a secondary server in LBRM. Receivers in LBRM can recover errors through a local secondary server instead of going to the primary server if the secondary server has a reliable copy. [4]

SRM is hindered by the fact that it multicasts its NACKs and retransmits to every group member. In an extension to SRM [11], Floyd et. al. are exploring "local recovery" in which the protocol tries to learn the extent of lost packets, and then uses this information to limit the scope of multicasted NACKs so that the multicast just reaches past the extent of lossage. This approach seems promising, but is currently ongoing research.

[4] However, studies on packet loss indicate that many losses occur at the tail circuits from routers to gateways. In these cases, the secondary servers will not reliably receive the packet and will have to go to the primary server.

3.3.1 Error Recovery Latency

The analysis in the previous section suggests that LBRM is much more scalable than SRM, and this is true when only considering message costs. However, the story is somewhat different when considering error recovery latency.

LBRM handles NACKs using fixed logging servers, and receivers reach the servers using a fixed reverse path tree. Placing fixed logging servers at each subdomain optimizes for localized losses. As long as all secondary servers reliably receive a multicast, any NACK can be satisfied quickly by a local secondary server. However, if a packet is not received by a secondary server, any NACK from its local domain must wait an RTT to the primary server, even if a secondary server nearby (perhaps even a single hop away) has a reliable copy of the packet. This will cause the error recovery latency to increase as the extent of group members grows.

The distance between the receivers and the primary server in LBRM also negatively impacts its NACK suppression methodology. LBRM uses statistical acknowledgment to trigger a multicasted retransmit thereby suppressing an implosion of NACKs. Like the RTT suffered by unicast recovery, there is an RTT suffered in statistical acknowledgment: the secondary server replies to the source as a designated acker, and then, if the source determines a multicast is appropriate, the multicast makes its way from the source back to the secondary server. The problem is that even if the source decides not to multicast the retransmit, receivers must give the source the opportunity to do so; otherwise, the multicast from the source will be too late to suppress an implosion of NACKs. In the case that recovery is accomplished via unicast, there is an additional RTT for the unicasted NACK and retransmit. Therefore, recovery can take up to 2 RTTs between the secondary servers and the primary server.

SRM has the potential for better scalability with respect to error recovery latency because it multicasts NACKs, and allows any group member to provide the retransmit. In theory, SRM has an error recovery time of 1 RTT between the receiver and the *nearest* reliable copy. However, this assumes that NACKs and retransmits can be issued without delay. Unfortunately, SRM must be very careful about multiple NACKs and retransmits since each is a multicast to the entire group. SRM uses randomized delay to desynchronize multiple NACKs (retransmits) in the hope that the first NACK (retransmit) will suppress all others. Since delay is inserted both before a NACK and a retransmit is issued, error recovery time can be significantly increased.

3.4 A Novel Network-Level Solution for Multicast NACK Suppression

From the discussion in Section 3.3, several problems related to supporting reliable multicast have been identified. One difficult problem is NACK suppression. LBRM does not adequately solve NACK suppression because it requires an RTT between secondary servers and the source before error detection can occur. SRM does not adequately solve NACK suppression because it inserts random delays to desynchronize NACKS, and even then, desynchronization is imperfect and multiple NACKs occur.

We claim the difficulty in current schemes to handle NACK suppression stems from the solutions using only the endpoints (i.e. the source and the receivers). The fundamental problem with an endpoint-based solution is that receivers who want to issue NACKs must insert delay in order to figure out when it's "o.k." to issue the NACK. This limits the scalability of error recovery time.

We propose providing support for NACK suppression at the network level. We believe a receiver should be allowed to issue a NACK immediately upon detection of a lost packet. It is then the responsibility of the network to identify redundant NACKs as soon as possible, and drop them in order to prevent their

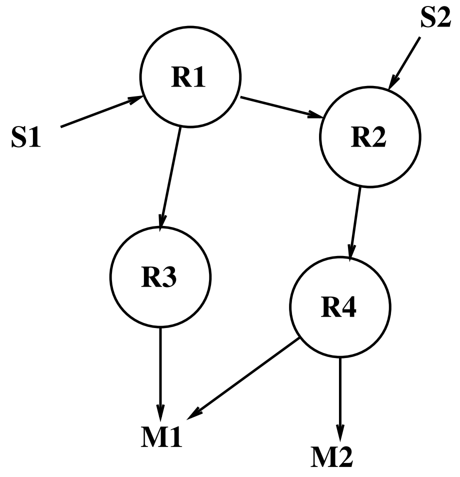

Figure 7: Suppressing a multicasted NACK. "S" nodes are sources, "R" nodes are routers, and "M" nodes are group members.

```
LINK_SET = empty;
Foreach incoming multicast packet {
    Determine outgoing links;
    Foreach outgoing link i {
        S = all members reached;
        if (S ⊂ LINK_SET) {
            Drop packet;
        } else {
            Forward packet through outgoing link i;
            LINK_SET = LINK_SET U S;
        }
}
```

Figure 8: Algorithm for suppressing multicast NACKs.

further consumption of network bandwidth. Our technique is motivated by (but is very different from) combining (or gathering) that has been previously proposed in [12] and [22]. In the following, we discuss a technique that will suppress multicasted NACKs; this technique would benefit the SRM protocol.

All NACKs are given tags. These tags appear in the network-level header of the NACK message. NACKs for the same lost packet have the same tag value. Switches in the network examine this tag value and maintain a tag cache of recently seen tags. Each time a message arrives at the switch, the tag is extracted, and the tag cache is probed. Based on the contents of the tag cache, the switch decides whether to forward the packet, or to drop the packet. A timeout mechanism can be employed for cache entries so that tag values can be recycled. Notice that different NACKs, whether they are new NACKs or retransmissions of lost NACKs, must use different tag values. Old tag values cannot be reused until it is certain that these tags have timed out in all the tag caches; otherwise, NACKs will be dropped incorrectly.

Suppression of multicasted NACKs is tricky, especially if shortest-path trees are used. Figure 7 illustrates the difficulty. Two sources, S1 and S2, multicast a NACK simultaneously to members M1 and M2. The shortest path to M2 from both S1 and S2 pass through router R2; however, the shortest path to M1 for S1 is through router R3, while for S2, the shortest path is through router R4. At router R2, there is an opportunity to drop a multicast packet. If S2's multicast packet arrives first followed by S1's multicast packet, it is safe to drop S1's packet since S2's packet will reach both M1 and M2. However, if instead S1's multicast packet arrives first, it is not clear that S2's multicast packet can be dropped. Doing so should be allowed only if it is known that S1's packet to M1 is not dropped at router R3 (due to, for instance, a multicast from a third source not shown).

To safely drop a multicast NACK, a router must determine that all members in the subtree reached by the NACK will receive the NACK from another multicast that has not been dropped. Figure 8 presents a conservative algorithm for suppressing multicast NACKs. The algorithm assumes each router has global membership information. In the algorithm, each router keeps track of all the multicasts that have passed through the router without being dropped, and records the group members reached by these previous multicasts. For each new multicast that arrives, it considers each outgoing link the multicast packet is to be copied and forwarded along. For a particular outgoing link, if the members reached through the link are covered by previous NACKs seen by the router, the NACK is dropped; otherwise, it is forwarded. Because the algorithm relies on routers having group information, it can be implemented by a link-state routing algorithm, but not by a distance vector routing algorithm.

3.4.1 Open Questions

In proposing network-level support for NACK suppression, we have created many open questions. We briefly discuss some of them here and suggest that they are directions for future research:

- The effectiveness of suppressing multicasted NACKs may be limited because routers must be conservative about which packets can be dropped. Future research can investigate the existence of less conservative approaches. Also, the proposed algorithm relies on link-state routing; the idea may not be practical unless a distance vector version is developed.
- The use of network-level tags to identify redundant packets poses the problem of maintaining the tags. How do hosts in the same group agree on a set of tag values? How are tags recycled so that a sufficiently small number can be used thereby not imposing too large a storage requirement in network-level headers?
- Are there other network-level mechanisms that can benefit reliable multicast in addition to NACK suppression? One direction that may have promise is caching of packets in the routers. This extends the benefit of getting a retransmit from a nearby host provided by SRM. In a network that can cache packets, a retransmit can be fulfilled by a router, not just a host. This approach can benefit from the emerging Active Networks research [24, 25, 30].
- Support for network-level mechanisms like NACK suppression can pose a problem in high-bandwidth routers. It has been proposed to support multicast routing in fast packet switching by building copy networks coupled with point-to-point networks in order to handle multicasting at extremely high rates [18, 33, 16, 26]. Integrating network-level support such as NACK suppression into high performance switches that support multicast, while minimizing the impact on hardware complexity and clock rate is a challenge.

4 Conclusion

This paper considers the scalability of different unreliable and reliable multicast protocols. The paper presents a model that allows the direct comparison of unreliable multicast routing algorithms under a single consistent framework. Using the model to study the scalability of four existing unreliable multicast protocols, we find that group density, the join/leave frequency of members, and the flood duty of RPM are first-order effects, while the cost of different distribution trees is a second-order effect. In particular, a high join/leave frequency can negatively impact the MOSPF protocol when application bandwidth is low. However, a somewhat surprising result is that MOSPF is scalable in situations where data message cost dominates control message cost, such as high application bandwidth, or in the case that application bandwidth is moderate, if groups are dense. A high flood duty impacts the cost of RPM unless groups are dense so that the cost of a broadcast is similar to the cost of a multicast. When groups are sparse, the only way to control the cost of RPM is to limit the flood duty factor. CBT and PIM both scale well under our analyses, but bandwidth concentration effects, which our model does not consider, are expected to negatively impact CBT.

This paper also provides an analysis of two competing reliable multicast protocols, LBRM and SRM. When considering messaging cost, LBRM is more scalable because it allows a gradual increase in cost as group size or per-receiver failure rates increase. SRM suffers from the fact that when at least one receiver fails to receive a multicast packet reliably, it incurs the full cost of at least two multicasts. However, in terms of error recovery latency, SRM is more scalable as it in theory can recover in 1 RTT to the *nearest* reliable copy, whereas LBRM takes up to 2 RTTs to the primary server. The SRM approach is appealing because it makes use of local reliable copies. The LBRM approach leverages local assistance only within domains; between domains, a retransmit can only be fulfilled by the primary server. For SRM to be feasible, it must address the high message cost of multicasting each NACK and retransmit to the entire group. A local recovery technique to limit the scope of multicasts, which is the subject of ongoing research, seems promising.

A problem that has not been adequately addressed in reliable multicast is NACK suppression. We feel this stems from the fact that current solutions make use of only the endpoints. We propose to deal with NACK suppression at the network level where information about traffic patterns is available. The paper suggests techniques for suppressing multicasted NACKs. Further research is necessary to evaluate the effectiveness of this technique, the problem of tag maintenance, and the implementation of this technique in routers, especially those that are designed to handle high data rates.

References

[1] S. Armstrong, A. Freier, and K. Marzullo. Multicast Transport Protocol. RFC (Request for Comments) 1301, Network Working Group, February 1992.

[2] Tony Ballardie, Paul Francis, and Jon Crowcroft. Core Based Trees (CBT) An Architecture for Scalable Inter-Domain Multicast Routing. In *Proceedings of ACM SIGCOMM'93*, pages 85–95, San Francisco, 1993. ACM Press.

[3] Jo-Mei Chang and N. F. Maxemchuk. Reliable Broadcast Protocols. *ACM Transactions on Computer Systems*, 2(3):251–273, August 1984.

[4] D. Clark, M. Lambert, and L. Zhang. NETBLT: A High Throughput Transport Protocol. In *Proceedings of ACM SIGCOMM'87*, pages 353–359, Cambridge, MA, August 1987. ACM Press.

[5] Yogen K. Dalal and Robert M. Metcalfe. Reverse Path Forwarding of Broadcast Packets. *Communications of the ACM*, 21(12):1040–1048, December 1978.

[6] Stephen Deering. Host Extensions for IP Multicasting. RFC (Request for Comments) 1112, Stanford University, August 1989.

[7] Stephen Deering, Deborah Estrin, Dino Farinacci, Van Jacobson, Ching-Gung Liu, and Liming Wei. An Architecture for Wide-Area Multicast Routing. *ACM SIGCOMM Computer Communication*, 24(4):126–135, October 1994.

[8] Stephen E. Deering and David R. Cheriton. Multicast Routing in Datagram Internetworks and Extended LANs. *ACM Transactions on Computer Systems*, 8(2):85–110, May 1990.

[9] Hans Eriksson. MBONE: The Multicast Backbone. *Communications of the ACM*, 37(8):54–60, August 1994.

[10] Sally Floyd, Van Jacobson, Steven McCanne, Ching-Gung Liu, and Lixia Zhang. A Reliable Multicast Framework for Light-weight Sessions and Application Level Framing. In *Proceedings of ACM SIGCOMM'95*, pages 342–356, Cambridge, MA, August 1995. ACM Press.

[11] Sally Floyd, Van Jacobson, Steven McCanne, Ching-Gung Liu, and Lixia Zhang. A Reliable Multicast Framework for Light-weight Sessions and Application Level Framing, Extended Report. Url ftp://ftp.ee.lbl.gov/papers/wb.tech.ps.z, Lawrence Berkeley Laboratory, September 1995.

[12] A. Gottlieb, R. Grishman, C. P. Kruskal, K. P. McAuliffe, L. Rudolph, and M. Snir. The NYU Ultracomputer – Designing a MIMD Shared-Memory Parallel Machine. *IEEE Transactions on Computers*, C-32(2):175–189, February 1983.

[13] C. Hedrick. Routing Information Protocol. RFC (Request for Comments) 1058, Rutgers University, June 1988.

[14] Hugh W. Holbrook, Sandeep K. Singhal, and David Cheriton. Log-Based Receiver-Reliable Multicast for Distributed Interactive Simulation. *ACM SIGCOMM Computer Communication*, 25(4):328–341, October 1995.

[15] M. Frans Kaashoek, Andrew S. Tannenbaum, Susan Flynn Hummel, and Henri E. Bal. An Efficient Reliable Broadcast Protocol. *Operating Systems Review*, pages 5–17, October 1989.

[16] Tony T. Lee. Nonblocking Copy Networks for Multicast Packet Switching. *IEEE Journal on Selected Areas in Communications*, 6(9):1455–1467, December 1988.

[17] John M. McQuillan, Ira Richer, and Eric C. Rosen. The New Routing Algorithm for the ARPANET. *IEEE Transactions on Communications*, COM-28(5):711–719, May 1980.

[18] Paul S. Min, Manjunath V. Hegde, Hossein Saidi, and Alex Chandra. Nonblocking Copy Networks in Multi-Channel Switching. *IEEE Transactions on Networking*, 3(6):857–871, December 1995.

[19] John Moy. Multicast Routing Extensions for OSPF. *Communications of the ACM*, 37(8):61–114, August 1994.

[20] Sridhar Pingali, Don Towsley, and James F. Kurose. A Comparison of Sender-Initiated and Receiver-Initiated Reliable Multicast Protocols. In *Proceedings of SIGMETRICS '94*, pages 221–230, Santa Clara, CA, 1994. ACM Press.

[21] Jon Postel. Transmission Control Protocol–DARPA Internet Program Protocol Specification. RFC (Request for Comments) 793, DARPA, September 1981.

[22] Bala Rajagopalan. Reliability and Scaling Issues in Multicast Communication. In *Proceedings of ACM SIGCOMM'92*, pages 188–198, Baltimore, August 1992. ACM Press.

[23] S. Ramakrishnan and B. N. Jain. Periodic Polling Protocol for Multicast over LANs. In *Proceedings of INFOCOM '87*, pages 502–511. IEEE, April 1987.

[24] D. L. Tennenhouse, S. J. Garland, L. Shrira, and M. F. Kaashoek. From Internet to ActiveNet. Request for comments, Laboratory for Computer Science, MIT, January 1996.

[25] David L. Tennenhouse and David J. Wetherall. Towards an Active Network Architecture. To appear in Computer Communication Review, 1996.

[26] Jonathan S. Turner. Design of a Broadcast Packet Network. In *Proceedings of INFOCOM '86*, pages 667–675. IEEE, 1986.

[27] D. Waitzman, C. Partridge, and S. Deering. Distance Vector Multicast Routing Protocol. RFC (Request for Comments) 1075, BBN, November 1988.

[28] Bernard M. Waxman. Routing of Multipoint Connections. *IEEE Journal on Selected Areas in Communications*, 6(9), December 1988.

[29] Liming Wei and Deborah Estrin. The Trade-offs of Multicast Trees and Algorithms. USC-CS 93-560, University of Southern California, September 1993.

[30] David J. Wetherall and David L. Tennenhouse. Active Networks: A New Substrate for Global Applications. Submitted to the 7th ACM SIGOPS European Workshop, Connemara, Ireland, September 1996.

[31] Brian Whetten, Todd Montgomery, and Simon Kaplan. A High Performance Totally Ordered Multicast Protocol. Theory and Practice in Distributed Systems, Springer Verlag LCNS 938.

[32] M. Yajnik, J. Kurose, and D. Towsley. Packet Loss Correlation in the MBone Multicast Network: Experimental Measurements and Markov Chain Models. In *Proceedings of INFOCOM '95*. IEEE, 1995.

[33] Wen De Zhong, Yoshikuni Onozato, and Jaidev Kaniyil. A Copy Network with Shared Buffers for Large-Scale Multicast ATM Switching. *IEEE/ACM Transactions on Networking*, 1(2):157–165, April 1993.

A Model Derivation for RPM, CBT, and PIM

In Section 2.3.1, the model components C_{cntrl}, C_{tree}, and C_{nm} were derived for MOSPF protocol. In this section, we do the same for RPM, CBT, and PIM.

A.1 RPM

Each time a host joins a group, the gateway for that host sends a message along the reverse-path rooted at the sender. This message cancels the prune of the branch that leads to the gateway. Although this message only needs to travel as far as the first non-pruned router, we are pessimistic in this model, and attribute $\frac{1}{2}$ the diameter number of hops of the network, N. Because an acknowledgment is expected at the gateway, the total cost is $diam(N)$. When a host leaves the group, a similar action occurs, except the message calls for a prune back up to the first non-pruned router.

$$C_{cntrl}^{RPM} = diam(N) \tag{8}$$

RPM uses shortest-path trees for distribution of unreliable multicast packets. Therefore, the cost of distribution of data messages to members is $C_{SPT}(N, G)$.

$$C_{tree}^{RPM} = C_{SPT}(N, G) \tag{9}$$

RPM avoids delivering packets down subtrees where there are no members by pruning. However, the prune entries in each router's routing table time out occasionally, and the entire network is flooded by the multicast (*i.e.*, a broadcast) momentarily until the unneeded branches are pruned back. This is necessary for group members to find new senders. If there are N routers in a network, then the broadcast across this network using Truncated Reverse-Path Broadcast (TRPB) [8] is N because there is one message that arrives to each router. The excess messages, *i.e.* the messages that do not reach group members in the TRPB, is simply the difference between the number of TRPB messages, and the number of messages in the pruned shortest-path tree. Our model assumes that RPM flips back and forth between a TRPB phase and an SPT phase, and that the fraction of time spent in the TRPB phase is $flood_{duty}$.

$$C_{nm}^{RPM} = (N - C_{SPT}(N, G)) flood_{duty} \tag{10}$$

A.2 CBT

Each time a host joins a group in CBT, it sends a message up the reverse-path core-based tree to add itself. Normally, this message only needs to go as far as the first non-pruned router. We are pessimistic in our model and assume that such a message traverses $\frac{1}{2}$ the diameter of the network, N. The same cost is incurred by an ACK that is sent back to the joining host.

$$C_{cntrl}^{CBT} = diam(N) \tag{11}$$

CBT uses core-based trees for distributing unreliable multicast packets. The cost of this tree is $C_{CBT}(N, G)$. We compute $C_{CBT}(N, G)$ in a similar way as $C_{SPT}(N, G)$. Given a network topology and a group size, we randomly place the group members. Then we find the optimal-delay core by considering all routers in the network and computing the average delay over all possible senders in the group. We do this for several random placements of the group. $C_{CBT}(N, G)$ is the average over all computed optimal-delay CBTs.

$$C_{tree}^{CBT} = C_{CBT}(N, G) \tag{12}$$

Core-based trees never deliver messages down subtrees that do not lead to group members.

$$C_{nm}^{CBT} = 0 \tag{13}$$

A.3 PIM

Each time a host joins a group in PIM, it sends a join up towards the core, or rendezvous point. This is similar to the (pessimistic) cost in CBT, $\frac{1}{2}$ the diameter of the network. Additionally, after the host finds a sender and it decides it wants to switch to a shortest-path tree, it sends another join message towards the sender, and a prune message towards the rendezvous point. We attribute another $\frac{1}{2}diam(N)$ cost to each of these operations.

$$C_{cntrl}^{PIM} = \frac{3}{2}diam(N) \tag{14}$$

We assume that in the steady-state, PIM uses shortest-path trees to deliver multicast messages. Therefore, the cost of data messages destined for group members is $C_{SPT}(N, G)$.

$$C_{tree}^{PIM} = C_{SPT}(N, G) \tag{15}$$

When using a shortest-path tree, PIM never delivers messages down subtrees that do not lead to group members. However, PIM sources always have to send to the rendezvous point, regardless of whether group members receive data through the shared tree rooted at the rendezvous point, or a separate shortest-path tree. Thus, it is possible for messages destined for the rendezvous point never to reach a group member. We regard this as a negligible cost.

$$C_{nm}^{PIM} = 0 \tag{16}$$