

LMS: A Router Assisted Scheme for Reliable Multicast

Christos Papadopoulos

University of Southern California
christos@isi.edu

*Guru Parulkar*¹

Washington University
guru@arl.wustl.edu

George Varghese

University of California at San Diego
varghese@ucsd.edu

Abstract

Building on the success of unicast IP, IP Multicast adopted a simple, open, best-effort delivery model with broadcast-within-a-group semantics. Despite several years of effort, the research community has been unable to produce scalable and reliable end-to-end transport on top of this model. Proposed solutions are either not flexible or incur high control overhead.

Many of these problems can be significantly simplified and perhaps solved with network assistance, an approach which had been dismissed until recently due to concerns about violating the end-to-end argument. Such violation occurs when network elements perform transport level operations. We claim that network assistance is possible without violating the end-to-end argument. We present Light-weight Multicast Services (LMS), which enhance the existing multicast model with simple forwarding services that do not require transport level actions. Using LMS, routers tag and steer transport-level control packets to selected endpoints, thus preserving the location advantage of the routers while pushing transport processing to the edges.

LMS facilitates very efficient solutions (compared to pure end-to-end schemes) to problems like scalable reliable multicast. We show that shifting the processing from routers to endpoints has minimal impact on efficiency (compared to the idealized transport-aware network solution), while offering significant gains in terms of scalability, reduced application complexity and ease of deployment. What distinguishes LMS from other scheme is its ability to easily handle dynamic groups and the minimal overhead it incurs at the network and the receivers.

1. Introduction

The Internet architecture is largely responsible for the undisputed success of the Internet today. At the core of this success lies the simplicity and elegance of the IP protocol whose design was guided by the end-to-end principle[12]. Internet architects realized that by foregoing the wire-like robustness of traditional networks (like the telephone network) and adding intelligence at the endpoints to compensate, a network can not only be built on a much simpler, cheaper and highly scalable infrastructure but can also adopt a very flexible service model well-suited for a wide range of applications.

1. Currently on leave of absence to Cisco Systems.

The Internet architecture is based on the so called *best-effort* service model. The interaction of store-and-forward packet forwarding, finite buffering and bursty data sources occasionally causes congestion and loss in the Internet. Applications requiring better than best-effort reliability must counteract loss by enriching end-to-end transmission with *error control*. Error control is the component of a communication protocol responsible for detecting and recovering loss. Error control has received wide attention in literature and good references include [1,2].

Internet Multicast debuted about ten years ago[5] and was welcomed as a natural extension and enhancement of the unicast model. Multicast provides great bandwidth savings compared to unicast when the same message must be delivered to a large number of recipients. Multicast is a very powerful communication service because it allows a single transmitter to reach an unlimited number of receivers in a very efficient and scalable manner. Many multicast applications exist today including streaming media, distance learning, Internet radio and television, distributed interactive simulation, file transfer, software updates and much more. Leveraging on the architectural success of the unicast model, Internet multicast adopted a best-effort broadcast-within-a-group service, akin to a radio channel: anyone can tune in and listen to transmissions by anyone transmitting on that channel. For simplicity and scalability, multicast groups remain anonymous, meaning that the network does not maintain state about receiver population and identity. Thus, similar to unicast the architecture provides a simple network delivery service on top of which richer semantics can be built.

Unfortunately, the end-to-end model which was so successful in the unicast case has proven much harder to apply to multicast. As a result, despite the vigorous promotion of multicast by both the research community and industry the Internet Service Providers (ISPs) and consequently the users have not yet embraced the service. Many reasons have been cited [46,47]. The most frequently mentioned are, lack of viable interdomain routing protocols, lack of capability to set-up complex peering relationships between autonomous systems, lack of scalable address allocation schemes, limited address space, difficulty with billing, and the lack of a general reliable transport service analogous to TCP.

Admittedly, the existence of a general multicast transport service that satisfies a large class of applications seems highly improbable. Factors like the heterogeneity of receivers and networks, diverse application requirements, and the dynamic nature of groups among others, preclude the development of a general transport service of reasonable complexity. Instead, the research community has recently focused towards developing and standardizing a set of multicast building blocks [48] that can be combined to compose a service that satisfies requirements of specific applications.

Recently, several proposals have emerged advocating changes to the current multicast model. EXPRESS[44] proposes a model where only one source is allowed to transmit and requires receivers to explicitly indicate their intention to join a group. The resulting multicast tree is routed at the source, which solves the rendezvous problem, and the explicit join abandons anonymity to allow access controls to enable services like authorization, authentication and billing. EXPRESS is geared towards highly populated one-to-many channels like Internet TV. Similar to EXPRESS, Simple Multicast [45] proposes to solve the rendezvous problem by including the address of the core or rendezvous point in the address of the group. Unlike EXPRESS, SM creates bidirectional shared trees which can handle many senders efficiently. Both approaches solve the problem of scarce multicast address space by using addresses that concatenate the source or the core address with the multicast address, essentially allowing 24 bits of multicast addressing *per router*. It is important to realize that neither EXPRESS nor Simple Multicast are meant to completely replace the existing service model; rather they are geared towards specific classes of applications where the existing model is less suitable.

In this paper, we address a different problem, not addressed by either EXPRESS or Simple Multicast, but essential for the ubiquitous deployment of multicast: providing a general and scalable reliable multicast service. Reliable multicast is desirable in many applications including file transfer, software updates, web cache updates, stock quote distribution and media archiving. Attempts to provide such a service within the current multicast model have proven elusive. One reason is the broadcast-within-a-group nature of the existing service model, whereas packet loss has localized effects. We therefore propose to extend the model with a set of **Light-weight Multicast Services (LMS)**, which extend router forwarding to allow localized operations which enables solutions that are highly scalable and far less complex than current solutions.

The broadcast nature of the current IP Multicast service model[5] makes it hard to perform reliable data transfer in a scalable and efficient manner. Traditional error control mechanisms designed for point-to-point applications (e.g., TCP) either break down completely or offer poor performance. The difficulties arise because the IP Multicast model requires that all messages sent to a multicast address reach all receivers subscribing to that address. This model is not well-suited for data recovery because losses in a multicast environment are mostly local, i.e., they affect only part of the multicast tree. Attempting to recover data lost locally by global means leads to several problems, including the following:

- **Implosion:** Implosion is a problem that occurs when the loss of a packet triggers simultaneous messages (requests and/or retransmissions) from a large number of receivers. In large multicast groups, these messages will swamp the sender or receivers, or even the network.
- **Exposure:** Exposure is a problem that occurs when recovery-related messages reach receivers which have not experienced loss. This is mainly due to the lack of “fine-grain” multicast in the existing model. Exposure wastes both network and endsystem resources.
- **Recovery latency:** The latency experienced by a member from the instant a loss is detected until a reply is received. Latency has significant implications in application utility and the amount of buffering required for retransmission.
- **Adaptability** to dynamic membership changes: This is a measure of how the efficiency (in terms of loss of service, duplicate messages and added processing and/or latency) of error recovery is affected by changes in the group topology and membership. In large dynamic multicast groups receivers may join or leave at random intervals. Thus error control mechanisms that make assumptions about receiver population and/or location are not suitable for such groups.

Current end-to-end solutions solve some, but not all of the above problems. For example, SRM[3] solves implosion at the expense of latency and exposure. RMTP[4] solves implosion, exposure and latency, but does not adapt well to dynamic membership changes. TMTP[14] adapts to changing group membership but uses complex heuristics.

LMS is motivated by the observation that current multicast solutions are significantly more complex than unicast solutions not only in terms of signalling but also in terms of state (especially topology-related state). The challenge, however, is to enhance the current model by adding minimal complexity to the network while maximizing the gain. We believe LMS meets this challenge. The key feature of LMS is the separation of the forwarding and error control components; the forwarding component is assigned to the routers (where it can be implemented most efficiently), while the error control component stays at the receivers (thus preserving the end-to-end principle).

Briefly, LMS works as follows: first, the routers create an implicit receiver hierarchy by selecting one of their downstream interfaces as the parent of the remaining downstream interfaces (the upstream interface is the one leading to the sender). Second, when detecting loss all receivers immediately multicast requests which are steered by routers to the parent interface. The exception is the request from the parent interface, which is forwarded upstream. This allows only one request to escape up to the higher level. Third, before

steering requests to the parent interface, a router stamps its address in passing requests; we call this router the *turning point*; it identifies the root of the subtree that sent the request. Since the turning point is carried in the request, routers need not remember any state. Finally, as requests climb up, some parent which has the required data will receive the request. That parent provides the turning point router with a retransmission which the router multicasts to the appropriate subtree.

Note how our scheme addresses all problems listed earlier: implosion and exposure are addressed by routers helping receivers create a virtual recovery tree, localizing recovery between parents and children. The tree adapts instantly to both membership and routing changes since routers ensure that the recovery tree always tracks the multicast routing tree. Recovery latency is minimized by adopting local recovery and having requests and retransmissions sent with no delay. Finally, the separation of forwarding and error control eliminates all topology state from the receivers (e.g., round-trip-time, back-off timers, explicit parent/child assignment), along with the associated signalling overhead.

It is important to understand the broader impact of our research on multicast. We have shown for the first time how to decompose multicast error control into two sets of components, namely forwarding and error control, and have shown that how this separation brings significant gains in performance and scalability while preserving the end-to-end principle. This is a promising approach which provides us with a vantage point from where to address other difficult multicast problems such as congestion control and more.

An expanded version of the work in this paper can be found in [49]. An earlier version of this work was published in [9]. The paper is structured as follows. In Section 2 we describe LMS. Section 3 presents our simulation results, including comparison between LMS and two other prominent schemes, namely SRM[3] and PGM[20]. In Section 4 we evaluate our implementation of LMS in the kernel of NetBSD Unix. In Section 5 we discuss related work. Finally, Section 6 concludes this paper. The Appendix includes further protocol details.

2. LMS: Light-weight Multicast Services

Light-weight Multicast Services (LMS) is a set of services provided by routers to greatly simplify the solutions to basic multicast transport problems. We emphasize that even though LMS was motivated by the problem of scalable multicast error control, LMS itself is not tied to a specific problem. We regard LMS as a set of general services that *facilitate* efficient solutions to difficult multicast problems. LMS enhances the existing IP multicast model with new *forwarding* functionality. By restricting its scope to forwarding, LMS avoids examining higher layers and thus does not violate the end-to-end principle. In this section we focus on the application of LMS to reliable multicast; later we propose other applications of LMS.

We begin by motivating the need to enhance the forwarding functionality of the current multicast service model. To do so we present an idealized error recovery scheme, shown Figure 1. Assume that a subset of receivers have just experienced loss when a packet on link L (marked with “X”) was dropped. Assume that the source is located at a distance from L ; there exists, however, an adjacent receiver (shaded in dark) that has received the data correctly and we would like to recover lost data from there rather than going back to the source. Let us call this receiver the *replier*. Let us also call the receiver who sends a request for the loss the *requestor*. We would like the requestor to be the closest receiver *downstream* the loss and the replier to be the closest receiver *upstream* the loss, because their proximity to the loss minimizes *recovery latency*. Recovery is initiated by the requestor by sending a NACK upstream (perhaps after detecting a gap in the sequence number) while all other downstream receivers remain silent, thus eliminating NACK implosion. The NACK is forwarded to the replier only, remaining invisible to other receivers upstream. Finally, the replier sends a retransmission via *constrained multicast*, such that begins at link L and travels downstream;

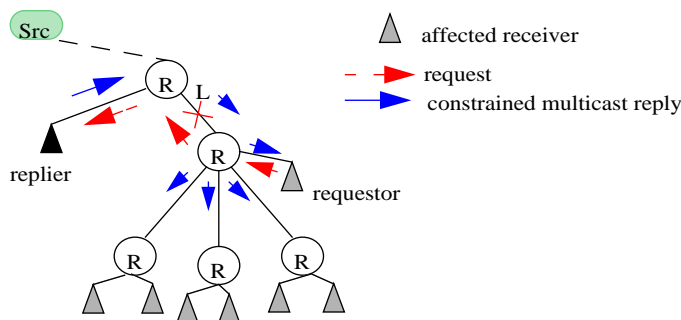


Figure 1: An imaginary, efficient recovery scenario

the retransmission rides the existing multicast subtree and is delivered to all receivers downstream of L (which happen to be the ones that missed the original packet).

We claim that this recovery scheme exhibits a near-optimal behavior:

- Only one NACK is generated by the receiver closest to the loss. A single NACK per lost packet is both necessary and sufficient to initiate recovery (assuming no NACK loss).
- Only one retransmission is generated by the receiver immediately above the loss. Again, assuming no further loss of retransmissions, a single retransmission is both necessary and sufficient to repair the loss.
- Only affected receivers receive the retransmission, which eliminates exposure. Utilizing the underlying multicast tree results in good efficiency (in terms of latency and bandwidth) in delivering the retransmission¹.
- By initiating NACKs and retransmissions immediately (no back-off timers) and recovering from the receiver closest to loss, recovery latency is kept at a minimum.

Unfortunately, however desirable it might be, the above scheme cannot be realized within the current multicast model. Requests cannot be targeted to specific receivers due to anonymity, the location of loss is unknown, and constrained multicast is not possible.

2.1. The LMS Concept

The idealized scheme above could be easily implemented if routers performed transport layer operations. However, this is impractical because of the heavy processing on the routers and the violation of the end-to-end principle. Yet, a router-based solution is conceptually simple and quite elegant if it can automatically build an optimal hierarchy for request suppression, and initiate fine-grain multicast of retransmissions. However, hidden in the above statement is the following key observation: it is not the router's processing power that the scheme exploits, but the router's *location*. In other words, the key advantage of enlisting routers to assist in error recovery, is not to utilize their processing cycles, but their tight coupling to topology. Recall that our major objections with the router-based schemes emanate from having routers do transport level processing (as does PGM); but as we just observed, processing is not the main advantage of this scheme. Would it then be possible to move the processing away from the routers, but maintain the location advantage? Or to restate the question:

1. In some topologies, however, the unicast path from the replier to receivers may be shorter.

In the heavy-weight router based model, efficiency is not achieved due to harnessing the routers' processing power, but due to their location. Would it then be possible to move the processing away from the routers while retaining the location advantage and thus preserving the elegance and efficiency of the model?

The above question lies at the heart of LMS; it also defines what LMS is: a set of services whose purpose is to allow the migration of transport-level processing away from the routers while retaining the location information. LMS achieves this by allowing the processing to be moved from each¹ router to another entity, called a *surrogate*. The surrogate thus becomes responsible for performing transport-level tasks on behalf of the router. The conceptual transformation from the router hierarchy to the surrogate model is depicted in Figure 2.

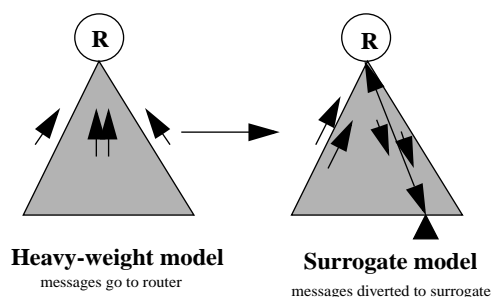


Figure 2: The LMS Concept

Who should the surrogate be then? We believe that the surrogates should be selected among the *receivers*. While expecting a receiver to do some work towards recovery is not unreasonable, selecting receivers as surrogates has significant advantages: (a) it pushes the load of recovery at the ends rather than concentrating it at the routers, (b) has great flexibility because any future changes can be done at the ends without affecting the network, and (c) all recovery operations remain within the transport layer, which avoids the layer violation of the earlier model. However, the cost is increased latency to steer information to a surrogate instead of processing it directly at the router. Also, surrogates may be slower than routers.

2.2. LMS Core Ideas

In order to migrate the processing from routers to surrogates we need to address the following questions:

- how does a router select a surrogate?
- how does a router redirect messages to its surrogate while preserving location information?
- how does the router relay messages from the surrogate?

We address each of these questions next, but first a word about our terminology: in the following sections the terms “surrogate,” “requestor” and “replier”, are used interchangeably. Both requestors and repliers are surrogates. Requestors are distinguished from repliers based on context, i.e., whether they are sending a request or a retransmission.

1. Here we are referring to routers that would have participated in recovery in the router hierarchy model.

2.2.1. Selecting a Replier (surrogate)

A router selects a replier for each source in a multicast group. Each router selects a replier as follows:

- if the router has two or more downstream links it selects one as the replier link¹;
- if the router has only one downstream link that becomes the replier link by default;
- if the source is directly attached to the router the source becomes the replier.

Figure 3 shows a possible router-replier allocation. The links leading to a replier are in bold. It is impor-

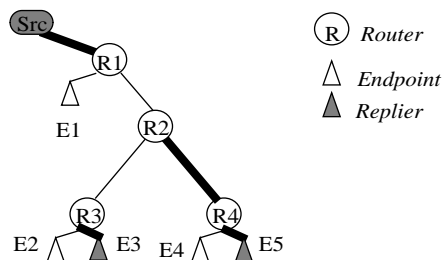


Figure 3: A possible replier allocation in LMS

tant to note that similar to data forwarding, a router only needs to know the replier *next hop*, not the actual replier. For example, router R2 selects R4 as the next hop knowing that it leads to *some* replier. This has some important advantages:

- If the replier changes, the change remains mostly local. For example, if R4 decides to switch to E4 as its replier (because E5 either left the group or crashed) R2 does not need to change its replier state.
- Receivers do not have to be notified that they have been selected as repliers. A receiver knows it has been selected if it receives a request. There is, however, no guarantee that the receiver will remain a replier for the next request.
- The replier state at the router is small. It is simply an identifier for the replier link.

In our discussion thus far, we did not specify which link the router selects as a replier link when there are two or more candidates. While the simplest solution would be to choose one at random, there are many reasons why we would prefer the receivers to control the replier selection. For example, some receivers may be better suited to act as surrogates because they are located on fast machines, or machines with larger memory. Therefore, next we introduce a replier selection mechanism.

Replier Selection Mechanism

Receivers express their desire to become repliers by piggybacking information on the join request. We expect that all receivers will be willing to become repliers (data recovery can be achieved even if this is not true, but at reduced efficiency). Along with the join request, receivers communicate a *cost* of their appropriateness as repliers, and routers pick repliers based on the advertised cost. Receivers repeat their current cost

1. We will address the issue of *which* downstream link shortly.

with every IGMP membership refresh. This process allows coordination among receivers to control replier allocation at the routers.

Replier Selection Space

LMS provides only a delivery mechanism for replier cost, meaning that the cost semantics are transparent to LMS. Routers make decisions based on simple comparisons. For example, groups wishing to minimize latency may use RTT as the advertised cost; others may use loss rate, or a combination of several metrics based on performance and/or policy. LMS leaves the selection of appropriate cost metrics to the application.

2.3. Steering Messages to Repliers

Each request is multicast, which keeps receiver actions simple¹. Requests, however, are handled hop-by-hop by the routers. Initially, a request travels on the reverse path towards the source until an appropriate surrogate path is found and then towards a replier using the replier forwarding state. Routers can easily steer requests towards the source (and block them from going downstream) by using the existing $\langle S, G \rangle$ state, where S is the original source. The hop-by-hop forwarding requires routers to examine each request, which is done via the IP Router Alert option [17], included in every request.

2.3.1. Request Handling at the Routers

LMS avoids request implosion because each router allows only one request to escape upstream - the one coming from the replier link. All other requests are funneled into the replier link. This is accomplished by routers steering requests to repliers as depicted in Figure 4.

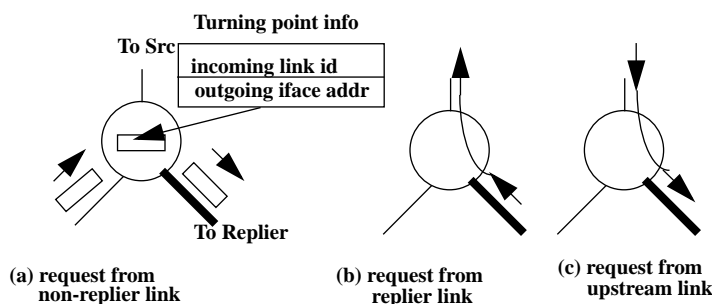


Figure 4: Request handling at a router

A request may arrive at a router from three possible directions:

1. **From a non-replier link:** note that this implies that the router has at least two downstream links. We call this router the *Turning Point* of the request. At the turning point the router turns requests around (recall that the request was traveling upstream until this point) and forwards them to the replier link. Before forwarding each request *and if the turning point field is empty*², the router adds turning point information to the packet, consisting of (a) an identifier for the link the request arrived on (the non-

1. PGM for example requires requests to be unicast to routers.

2. See later discussion on proxy directed multicast in section A.7 as an example of where a router may find the turning point field non-empty.

replier link) and (b) the address of the replier interface. Note that the turning point information globally identifies the root of the subtree that sent the request. We will see shortly where the turning point information is used.

2. **From the replier link:** when a request arrives from the replier link the router forwards it to the upstream link without touching the packet.
3. **From the upstream link:** when a request arrives from the upstream link the router forwards it to replier link unchanged.

The maximum number of requests diverted to the replier is bounded by the number of downstream links at the turning point. Assuming that routers do not have a large fan out, repliers will not experience implosion. For routers with a large fan out we propose a solution in section A.6 A fine but important point is that the replier that actually has the data (and will serve the request) receives at most one request for a particular packet loss.

2.3.2. Directed Multicast (DMCAST)

If a replier receives a request but does not have the requested data, the replier ignores the request since it must have sent a similar request of its own¹. Otherwise, the replier retransmits the data using a *Directed Multicast (DMCAST)*. This is the final service LMS adds to the routers; its purpose is to enable fine-grain multicast to eliminate exposure.

The operation of a DMCAST is shown in Figure 5. To perform a DMCAST the replier creates a multi-

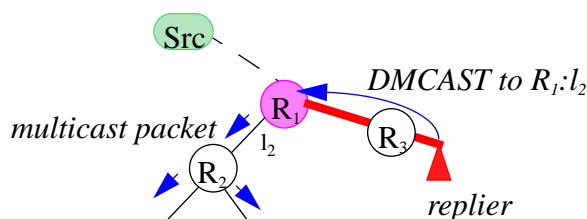


Figure 5: A Directed Multicast (DMCAST)

cast packet containing the requested data and addresses it to the group. An IP option is added containing the turning point link identifier obtained from the request. The replier then encapsulates the multicast packet in a unicast packet and sends it to the turning point router, whose address was again obtained from the request. When the turning point router receives the unicast packet, it decapsulates the multicast packet, strips the IP option and multicasts it on the specified link. From there the packet is forwarded like a regular multicast packet originating from the source.

DMCAST's leverage off the information inserted by routers at the turning point to achieve fine grain multicast and eliminate exposure. The turning point identifies the root of the loss subtree and a dmcast is able to target this point precisely, thus sending retransmissions only to interested receivers in the group.

1. See later discussion on what happens when replier does not have the data because its buffers were flushed.

2.3.3. LMS Summary

In summary, LMS employs three important concepts: replier selection, steering of requests to repliers and establishing turning points, and directed multicast. These concepts work together to enable receivers to construct an efficient recovery mechanism which is depicted in Figure 6 and summarized below:

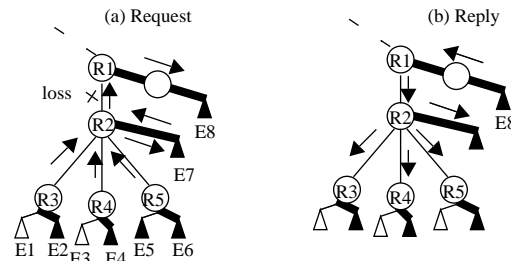


Figure 6: LMS Summary

Sending a request:

Replier links are in bold. Loss occurs on the link between R1 and R2. Endpoints E1 through E7 detect the loss. Then, the following events take place:

- E7 sends a request, which R2 forwards to R1 because E7 lies on R2's replier link.
- E1 sends a request which is forwarded by R3 to E2. Similarly, requests from E3 and E5 are forwarded to E4 and E6 by R4 and R5, respectively.
- The request from E2 is forwarded to R2, because E2 is on R3's replier link. Similarly, the requests from E4 and E6 are also forwarded to R2.
- R2 forwards requests from E2, E4 and E6, to E7, which ignores the requests since it does not have the data (but has requested it).
- The request from E7 reaches R1, which forwards it towards E8, which has the requested data.

Sending a Reply

Once E8 receives the request and determines that it has the requested data, it prepares a reply and sends it as follows:

- E8 creates a multicast message containing the reply. E8 encapsulates the message in a unicast message and sends it to R1 (the request's turning point).
- R1 decapsulates the multicast message and multicasts it on the link leading to R2.
- From that point on, all downstream routers and endpoints treat the reply as a regular multicast message coming from the source.

Note that LMS routers maintain no state other than the replier link identifier and cost, which is independent of the number of receivers. Routers need not remember anything about requests as they pass through. Routers are not even aware that these recovery messages, they simply forward messages according to the new forwarding rules just as they forward data packets according to the standard multicast rules. Thus, LMS imposes no per-packet state at the routers.

Since both data and LMS packets are forwarded using the same $\langle S, G \rangle$ state, the router forwarding state for single-sender groups is minimized. In contrast, a scheme like SRM can potentially create $\langle S, G \rangle$ state for each receiver, a significant overhead when the number of sources is small.

2.4. Problem: Exposure

Since there is always one replier, no duplicate replies are generated for the same loss. It is possible, however, that an unaffected receiver may be exposed to replies generated as a result of recovery initiated by affected receivers. An example is shown in Figure 7. In this example, a request from replier 1 reaches replier

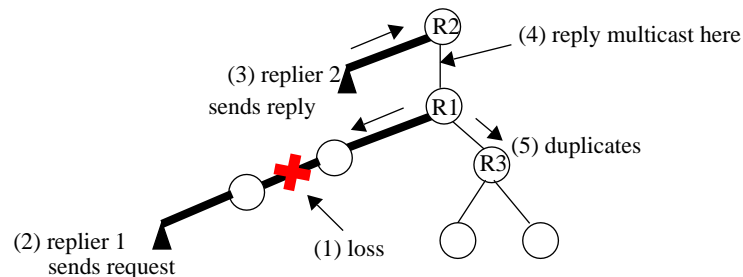


Figure 7: Exposure in LMS

2. In response to the request, Replier 2 sends a directed multicast to R2, which multicasts the reply on the downstream link leading to R1, causing duplicates on the branch towards R3.

Even though this problem does not inhibit recovery, it may lead to the “crying baby problem,” where excessive loss experienced in one branch causes duplicates at a large number of other receivers. With LMS this problem can be mitigated by using the cost field to select a replier that advertises the least loss. For example, R1 will select a replier from the right-hand-side branch if this branch experiences less loss, even though the replier on the left-hand-side branch may be closer.

2.4.1. Eliminating Exposure

As we have seen earlier, loss at a replier link may cause duplicate messages to reach some receivers. There is, however, a method to eliminate these duplicate messages, at the cost of adding an extra hop to the retransmission. This approach is shown in Figure 8. We do *not* use this enhancement in our simulations.

To eliminate exposure, each request specifies that the reply should be unicast to the requestor rather than the turning point. If the requestor receives any other requests, either while waiting for the reply or soon after, the requestor knows that this is a loss that affected more receivers than just itself. Therefore, it initiates a DMCAST to the remaining receivers. The requestor may choose to respond to each request with a separate DMCAST or initiate a single DMACAST to all downstream links at the turning point.

3. Simulation Results

We evaluated the performance of LMS using simulation and compared it with two other reliable multicast schemes, namely SRM [3] and PGM [20]. The reason for including SRM in this group is not to compare it with LMS and PGM, because such a comparison would be unfair. We include SRM for the simple purpose

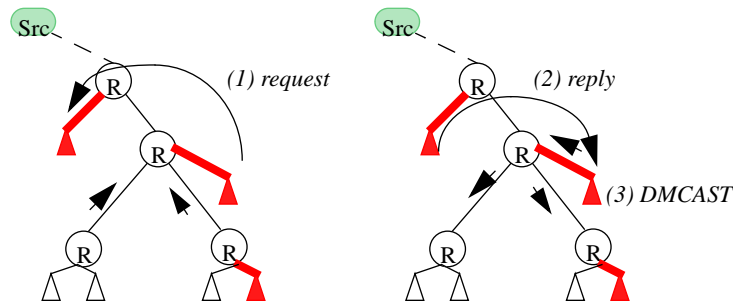


Figure 8: Two-step recovery eliminates exposure (not used in our simulations)

of demonstrating what is achievable with and without network support. The final decision of whether the benefits justify the deployment of router assistance in the Internet is beyond the scope of this paper. We merely hope to illuminate some of the trade-offs before making such a decision.

The chosen simulator is the UCB/LBNL/VINT Network Simulator - *ns*[21]. Most of the topologies were generated via GT-ITM[22]. Although not very large (about 200 nodes), the topologies generated were the largest possible given the memory in the machines at our disposal (64MB). We are confident, however, that our conclusions will be valid for larger topologies.

In order to avoid dealing with congestion¹, we used artificial packet drops. Similar to previous evaluations [3] we measured the overhead of recovery at each scheme *after* a loss occurred. For the same reason we modeled drops of original packets only, not retransmissions. We ran numerous simulations with a wide range of topologies and scenarios, using identical scenarios for all three schemes. We only simulated a single source sending data to many receivers. We do not expect our results to change with multiple sources.

3.1. Evaluation Metrics

While some studies have been carried out to attempt to characterize loss in the Internet [13, 23], their results have not been conclusive. Moreover, it is not clear that loss measurements done on the MBONE will be valid when multicast migrates to ISPs. Thus, to avoid making our own (most likely flawed) assumptions about future multicast networks, we limit our study to just two performance metrics: latency and duplicates. A similar set of metrics was used to evaluate SRM. We did not evaluate proposals for adding local recovery to SRM[43]. The following metrics were used: (a) *normalized recovery latency* (for all schemes), (b) *exposure* (for LMS), (c) *requests/repairs per drop* (for SRM), and (d) *repeated retransmissions per drop* (for PGM). Exposure and repeated retransmissions are new metrics. The definitions of these metrics appear in Figure 9.

Normalized recovery latency is defined as the latency a receiver experiences from the moment it detects a loss until the loss is recovered, divided by that receiver's round-trip time to the sender. Exposure applies to LMS and is defined as the average number of duplicate messages received by a receiver as a result of loss at some part of the multicast tree (which may or may not have affected the receiver). Exposure attempts to capture the degree of success of local recovery in LMS. It does not apply to PGM because PGM has very precise local recovery. For SRM we used the same metrics employed by the SRM designers. Repeated Retransmissions is a metric that applies to PGM. In PGM NACKs set up state as they travel towards the

¹. None of the three schemes under evaluation includes congestion control.

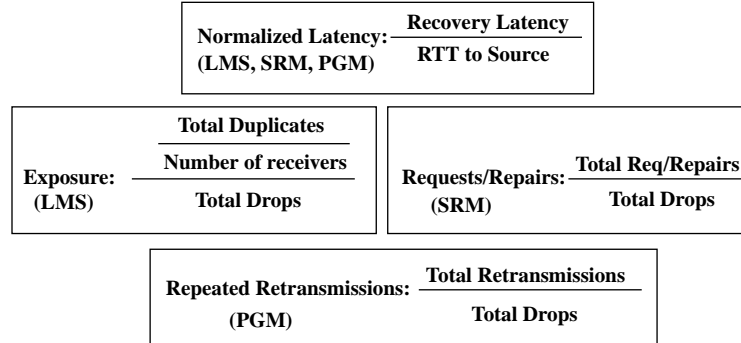


Figure 9: Evaluation metrics

source and retransmissions erase the state as they travel towards the receivers. In some topologies it is possible that retransmission state is wiped out too fast, before all links are grafted into the retransmission tree. In these cases, the sender must send multiple retransmissions to repair the loss at all receivers.

We used three types of topologies in our simulations: binary trees, random, and transit-stub (TS) topologies. Random and TS topologies were generated with the Georgia Tech Internet Topology Models (GT-ITM). The binary tree topology, while an unrealistic topology (i.e., unlikely to be encountered frequently or at a large scale in the Internet), was chosen because it represents a regular, easy to visualize topology. Binary trees are a difficult case for both randomized and hierarchical protocols: randomized protocols have difficulty selecting appropriate timer back-off values when the distance of all receivers from the source is approximately the same; hierarchical protocols have difficulties selecting appropriate helpers when all receivers are equally good (or bad) candidates. Thus, binary trees provide a degree of stress testing for both types of protocol, yet they are a topology that is easy to visualize.

3.2. Simulation Parameters

For binary trees we simulated trees of height ranging from 3 to 7 (8 to 128 receivers). For random and transit-stub simulations we used topologies containing up to 200 nodes (100 internal nodes and 100 receivers). We generated 10 random and 10 transit-stub topologies each containing 100 nodes. We run simulations with 5, 20 and 100 receivers, randomly distributed over the internal nodes. For each topology we ran 10 simulations each time with a different receiver allocation (generated by feeding a random seed to the random number generator). Thus, each plot is the result of 100 simulation runs. For random topologies the receiver placement was random on all internal nodes; for transit-stub topologies receiver placement was also random, but only on stub nodes.

As discussed earlier, the loss characteristics of future multicast networks are unknown. Thus, we did not attempt to make any assumptions about the loss characteristics (e.g., loss duration and location); instead, we limited our scenarios to a single packet loss; for loss location, (which has significant impact on performance), we chose to investigate the following three cases:

- **Loss at the source:** a packet is lost near the source such that none of the receivers gets it. This case tests the scheme's ability to control NACK implosion.
- **Loss at each receiver:** a packet is lost such that it affects only a single receiver. This case tests the scheme's ability to control exposure.

- **Loss at each link:** in this scenario, loss is moved from link to link during a single simulation run, until all links are covered. This is roughly equivalent to random loss where all links have equal loss probability, and is meant to test cases between the other two.

While we certainly do not claim that these cases represent real loss characteristics of future multicast networks, we believe that they provide enough information to attain a basic understanding of the behavior of these error control schemes. As we learn more about the loss characteristics of future multicast networks, updated loss models can be plugged into our simulations to obtain more precise results.

3.3. LMS Experiments: Static v.s. Dynamic Repliers

Unless otherwise stated, *the repliers in the LMS experiments were static*, selected at the beginning of the simulation. Despite the potentially severe performance limitations this entails, we kept the repliers static for two reasons: (a) without knowledge of the loss characteristics of the network, it is hard to devise an efficient replier adaptation scheme, and (b) we wanted to explore the performance of LMS with simple replier allocation. With static repliers, the results presented for LMS are certainly not the best LMS can achieve. This is especially true for exposure, where the replier selection is the most important factor governing performance. As our experiments will show, LMS performs very well even with static repliers.

3.3.1. Binary Trees

In the first experiment we used binary trees with height ranging from 3 (8 receivers) to 7 (128 receivers) and loss at the source. The results are shown in Figure 10. We plot the average, minimum and maximum

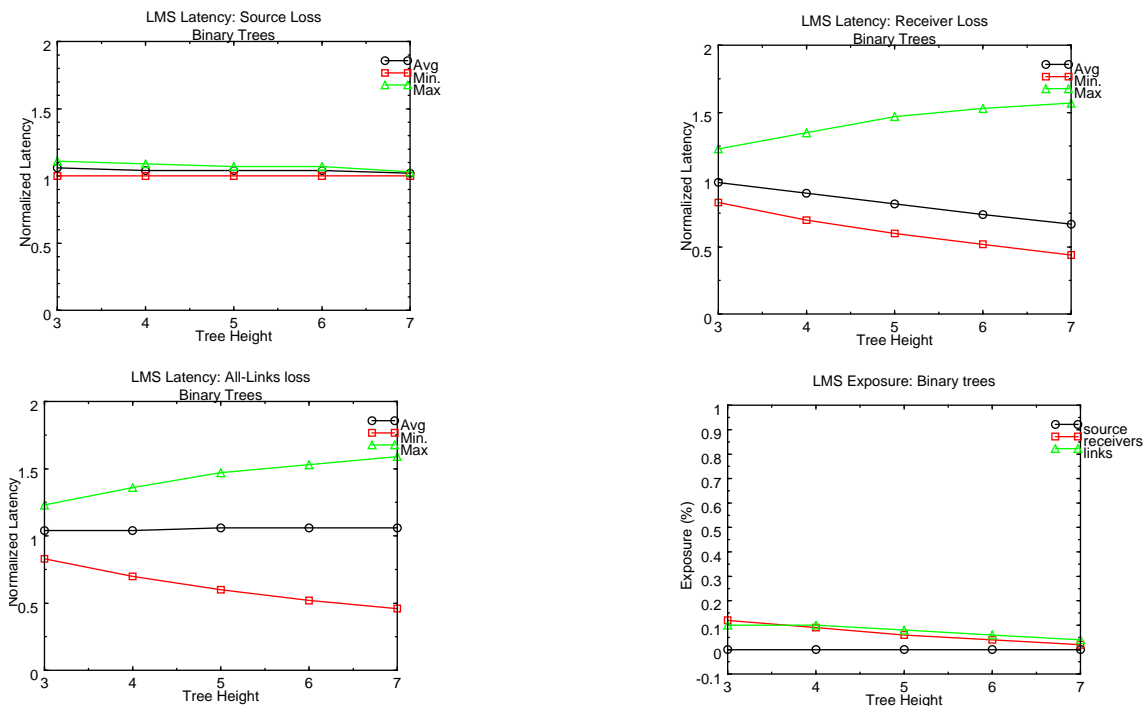


Figure 10: LMS with binary trees

recovery latency. We do not use error bars in these graphs because the results with binary trees are deterministic. The x-axis lists the five different topologies used in the experiment and the y-axis the normalized

recovery latency. We observe that since all receivers have the same RTT to the source, the recovery latency is close to 1. The minor deviations are caused by slight queueing time due to synchronized requests. Note that the recovery latency does not change much as the tree height is increased.

In the next experiment we simulate loss at the receivers. Here we observe that the average recovery latency decreases as the tree height increases, so for larger trees recovery gets faster. The maximum latency increases because loss at some receivers causes a NACK to propagate towards the source only to be turned around and delivered to another receiver instead. This causes a loss to be recovered from a receiver that happens to be further away from the requesting receiver than the source, thus increasing the recovery latency to beyond 1. Note, however, that in binary trees the normalized recovery latency can never exceed 2. The reason is as follows: the maximum distance (in number of hops) between any two receivers is given by the expression: $d = 2 \cdot (h - 1)$. As the tree height increases, the maximum normalized latency becomes:

$$\lim_{h \rightarrow \infty} \frac{2 \cdot (h - 1)}{h} = 2$$

In the third recovery latency experiment, the loss is moved around from link to link until all links are visited. As the tree height increases we observe that the average recovery latency remains unaffected and close to 1. The maximum latency increases as described earlier. The minimum latency decreases because as the tree gets taller the ratio of the distance to a neighbor to the distance to the source decreases.

In the last experiment with binary topologies we measure the exposure as the tree height increases. Recall that in these experiments repliers are kept static. We measured exposure for all loss cases, namely source, receivers and links. Note that the exposure when loss is at the source is zero because all receivers need the retransmission. For the remaining cases exposure starts out low (less than 15%) and decreases quickly as the height of the tree increases.

In summary, even though binary trees are a difficult topology for LMS due to lack of internal helpers, LMS still performs well. It recovers losses in about one RTT, and keeps exposure very low.

3.3.2. Random Topologies

As mentioned earlier, the random topologies we used were generated with GT-ITM. The edges were assigned from a uniform distribution with probability 0.1. We generated topologies consisting of 100 nodes and randomly assigned 100 receivers and a source to these nodes, bringing the total number of nodes to 201. Each topology was used for 10 runs, with receivers re-assigned before each run. With 10 topologies each experiment required 100 simulation runs.

As with binary trees, we first measured recovery latency. The first graph in Figure 11 shows results with loss at the source. The term R100-100 means “random graph with 100 nodes and 100 receivers.” From the figure we see that on the average recovery takes about 30-40% of the unicast RTT, which is significantly better than with binary trees. The reason is that in random topologies there are many helpers in the internal nodes. The maximum latency is again around 1, experienced by receivers that are close to the source.

In the next set of results we measure recovery latency with loss at the receivers. Here we observe that the average latency increases slightly to about 50%. The reason can be deduced by looking at maximum latency, which has increased to about 1.25. The increase is due to LMS selecting repliers that are located at larger distance than the source, as it happened with binary trees. With loss at all links, the results do not change significantly. The average latency again increases slightly to about 60% and minimum and maximum latencies are essentially unchanged.

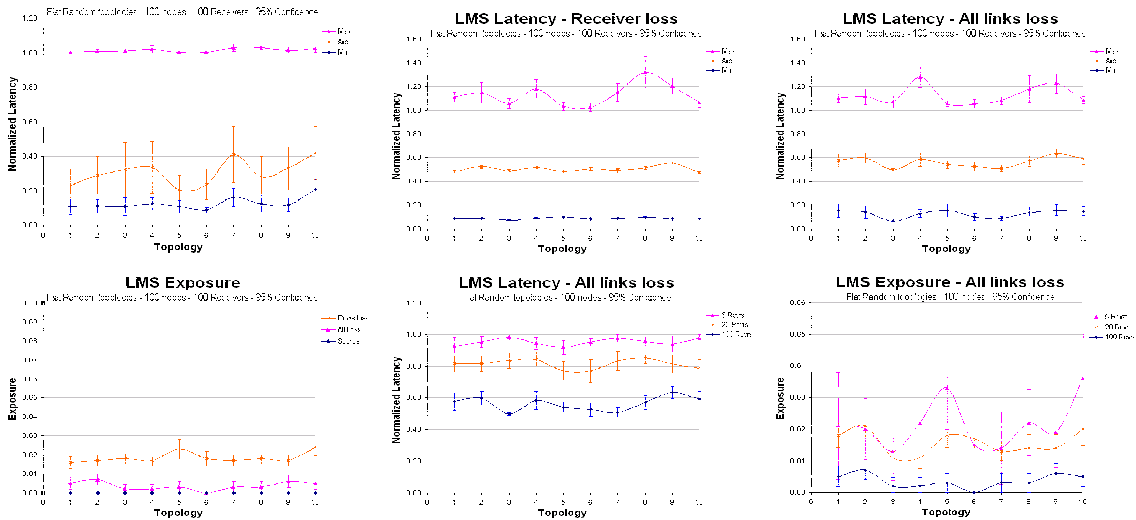


Figure 11: LMS with random topologies

The results for exposure show that exposure is very low, under 2% in most cases, and well below 3% in all cases. For loss at the source there is no exposure. The highest exposure occurs, as expected, when loss is at the receivers because a loss at a receiver acting as a replier may cause duplicates at other receivers. When loss is equally distributed between all links exposure remains very low, under 1%. As with latency exposure is much better with random graphs than with binary trees, confirming our initial claim that binary trees are a difficult topology for LMS.

Varying the number of receivers

In this experiment we test the performance of LMS with small sparse groups. We used the same topologies as in the previous experiments altering only the number of receivers. The results are shown in Figure 11. We plot simulation results with 5, 20 and 100 receivers, the latter being taken from the previous experiments. We used loss at all links and plot only the average latency. From the results we notice that the recovery latency is inversely proportional to the number of receivers. By definition, recovery latency is 1 when there is only one receiver, and gets progressively smaller as more receivers are added. The same applies to exposure: larger receiver population leads to less exposure. Note, however, that even with few receivers exposure is still very low (less than 4%).

The fact that performance improves as the group gets larger is good news. The reason performance improves is that more helpers are available to initiate and send retransmissions, which improves latency; more helpers also means that there is a better chance to find a helper better located to serve retransmissions without causing exposure. The observation that performance improves with larger groups is an interesting result which is actually not limited to LMS, but also applies to SRM and PGM in varying degrees. For example, while in SRM latency improves as the number of receivers increases, in order to reduce duplicates from the higher receiver population the timer back-off values may have to be further increased which can offset the latency gains. In PGM, the presence of more receivers will initiate retransmissions faster, but may worsen the repeated retransmissions problem. Recovery latency is the sum of the NACK propagation latency to the source plus the propagation latency of the retransmission to the receivers. The presence of more receivers may shorten the former, but not the latter. Also since PGM recovers from the source in most cases, the latency reduction will not be as great as with LMS.

3.3.3. Transit-Stub Topologies

In this section, we examine the performance of LMS with transit-stub topologies. While random topologies are a good approximation of dense, richly connected topologies, transit-stub topologies are a better approximation of the hierarchical structure of the Internet.

As with random topologies, we generated 10 topologies with 100 nodes each. The parameters fed to GT-ITM to generate the topologies are as follows: 1 top-level domain (the transit domain) with 4 transit nodes; each transit node had 3 transit-stub nodes; and each transit-stub node had 8 stub nodes. This brings the total number of nodes to $1 \times 4 \times (1 + 3 \times 8) = 100$ nodes. As with random topologies we randomly assigned 100 receivers, but in this case the receivers were assigned to stub nodes only. Unlike random topologies instead of simulating loss at the source and receivers, we opted to simulate loss at the different types of links. Thus, we studied loss at transit-transit, transit-stub, and stub-stub links. The motivation was that we were interested in finding out how loss at each type of link affects performance. However, as with random topologies we also produced results with loss at all links.

The results are shown in Figure 12. While latency remains at about 50% on the average when all links

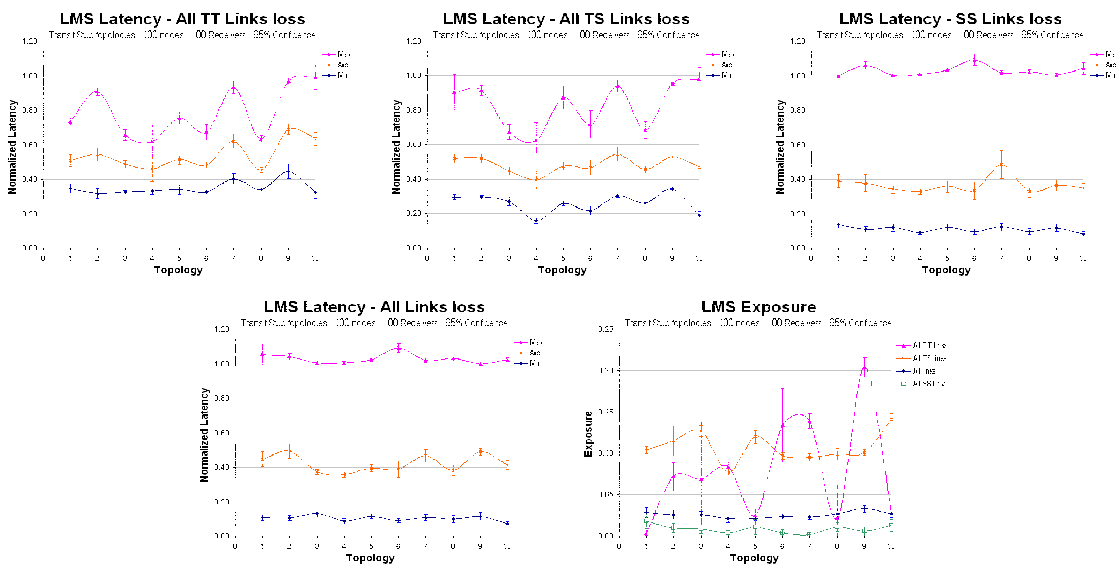


Figure 12: LMS transit-stub topologies

are lossy, latency is a bit higher for loss at the higher levels (transit-transit links) and less so at the middle level (transit-stub links). However, the difference is small and in general the results are on par with random topologies. The situation, however, is different with exposure. While exposure remains low with loss near the receivers (on stub-stub links), it increases significantly with loss at the higher levels (transit-transit and transit-stub), reaching peaks of 15-20%. While this is not alarmingly high, it seems to be highly dependent on topology and receiver allocation (for example topologies 0, 4, 7 and 9 have very low exposure whereas topologies 5, 6, and 8 have somewhat higher exposure).

3.4. SRM Experiments

SRM employs two clever global mechanisms to limit the number of messages generated, namely duplicate suppression and back-off timers. In SRM, recovery messages (requests and replies) are multicast to the entire group; receivers listen for recovery messages from other receivers before sending their own, and suppress their recovery messages if they would duplicate one already seen. The intended goal is to allow the multicast of only one recovery message. In order to increase the effectiveness of the suppression mechanism, especially in densely packed groups, the round-trip-time between receivers is artificially enlarged (for recovery messages only) with the addition of back-off delay. To improve performance, the added delay consists of a fixed and a random component, calculated separately at each receiver. The fixed component is based on the distance of the receiver to each sender, and the random component is based on the density of the receivers in the neighborhood. However, these components have to be re-calculated when group membership, topology, or network conditions change, meaning that SRM needs time to adapt to improve performance.

SRM has already been extensively studied via simulation and results have been reported elsewhere [3]. Our goal here was not to repeat or extend already published results, but to compare SRM on the same topologies used for LMS. However, we were not completely successful in achieving that goal. While we used the same topologies, we could not run SRM simulations with more than 20 receivers in the 100-node topologies. Attempting to use more receivers resulted in extremely long simulation runs and very high memory consumption. We briefly gained access to an UltraSparc with dual processors and over 1GB of RAM for SRM simulations, which took about 3 days to finish each set of 100 runs. The reason SRM simulations are so slow is that the SRM implementation in *ns* is done mostly in Tcl in order to maximize flexibility; unfortunately this came at the price of speed and memory consumption. We did not attempt to improve the simulation running time (which would require re-writing the simulation in C++).

The results we report are consistent with results presented previously, and thus we do not feel we would have contributed further to the understanding of SRM had we simulated larger topologies. In the following sections we present results from simulating SRM in random topologies only. As mentioned before, further study of SRM is beyond the scope of this work. The random topologies are the same used with LMS, but with only 20 receivers. We report results for normalized latency and the number of requests and replies generated for each lost packet.

Figure 13 shows the recovery latency for loss at the source, receivers and links. We note that on the average, SRM recovers from a loss in about 2 RTTs, or twice the unicast latency, with the maximum value being around 4 and the minimum around 1. We also note that the recovery latency is relatively uniform over all topologies. We believe that the reason is that the back-off timers absorb any differences that may arise due to a particular topology. In addition to being relatively insensitive to topology variations, SRM appears to also be insensitive to loss location. Thus, there are very small differences between results from loss at the source, receivers or links.

We also show the number of requests and replies generated on the average for each loss. For requests, the linear component in the back-off timers works well and keeps the number of requests low (a little above 1 for loss on all links). The number of requests are a bit higher when loss is at the source because all receivers compete in sending a request. The results, however, are significantly worse for replies where SRM may generate 4 -5 replies for each loss. For replies the linear component is less effective since the inter-receiver RTT is smaller in general than each receiver's RTT to the source.

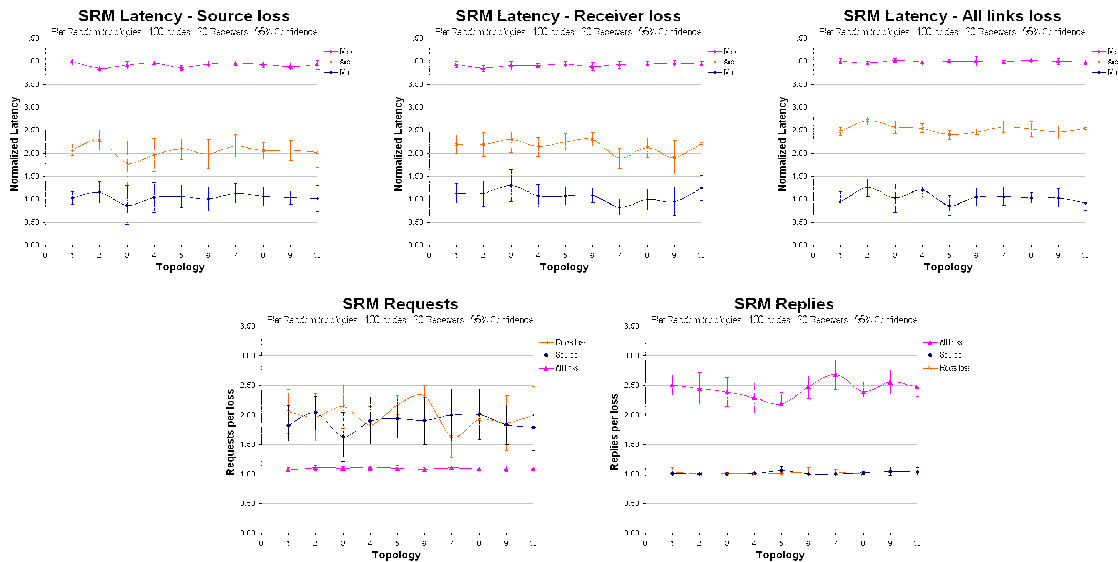


Figure 13: SRM with random topologies

3.5. PGM Experiments

PGM[20] is a reliable multicast protocol marketed by the router company Cisco. PGM is a network-assisted scheme, that unlike the schemes we described earlier, examines the transport layer and requires per-lost-packet state at the routers. PGM uses NAKs to create state at the routers that is used to avoid sending duplicate NAKs upstream and to guide retransmissions to receivers that requested them. In PGM, all retransmissions originate from the source. Provision is made for suitable receivers to act as Designated Local Retransmitters (DLRs). PGM in its current specification, faces the following problems:

- **Dangling NAK state:** if a NAK or RDATA is lost, previous NAK state is not discarded at the routers until the NAK state expires. Thus, when receivers that fail to receive RDATA timeout and send a NAK again, these NAKs are blocked by the routers. The reason is that PGM routers block duplicate NAKs from being propagated upstream while NAK state is present. Since NAK state is normally erased by passing RDATA which did not arrive, the state is still present until it times out. The NAK state expiration interval, however, is just a soft-state safeguard to eliminate stale state, and thus is typically large (several seconds). The solution proposed by the PGM designers is to make the NAK state permeable to one NAK after 1 second, thus limiting the amount of time a receiver's NAKs can be blocked.
- **Repeated retransmissions:** While PGM in its current specification guarantees that a retransmission will not reach a receiver that has not requested it, it does not guarantee that a single retransmission will cover all receivers that have requested a retransmission. In some topologies PGM may have to send the same retransmission multiple times to cover all receivers. The reason is that a receiver close to the loss may send a NAK and trigger a retransmission before NAKs from distant receivers have a chance to establish NAK state in downstream routers. Since NAK state is wiped out by RDATA, a NAK arriving at a router after RDATA went through will re-establish NAK state back to the source. We examine the impact of this problem in our simulations.

To the best of our knowledge, our simulation was the first simulation for PGM. Results from our simulation were first presented at the 4th Reliable Multicast Research Group meeting [24], where one of the

PGM designers from Cisco was present. There, we were informed that PGM was already at an advanced stage of implementation.

The topologies and parameters used for the PGM experiments are the same as with LMS and SRM, except that we did not have the memory limitations we experienced with SRM (the PGM simulation was written mostly in C++) and thus our simulations used 100 receivers, as in LMS. Our PGM simulation did not include all the features described in the PGM specification. However, we believe our simulation included enough of the PGM functionality to capture the basic operation of PGM. Thus, we believe that the results we present here are applicable to a full-fledged PGM implementation.

Figure 14 shows the PGM results. With loss at the source, average recovery takes about 80% of the uni-

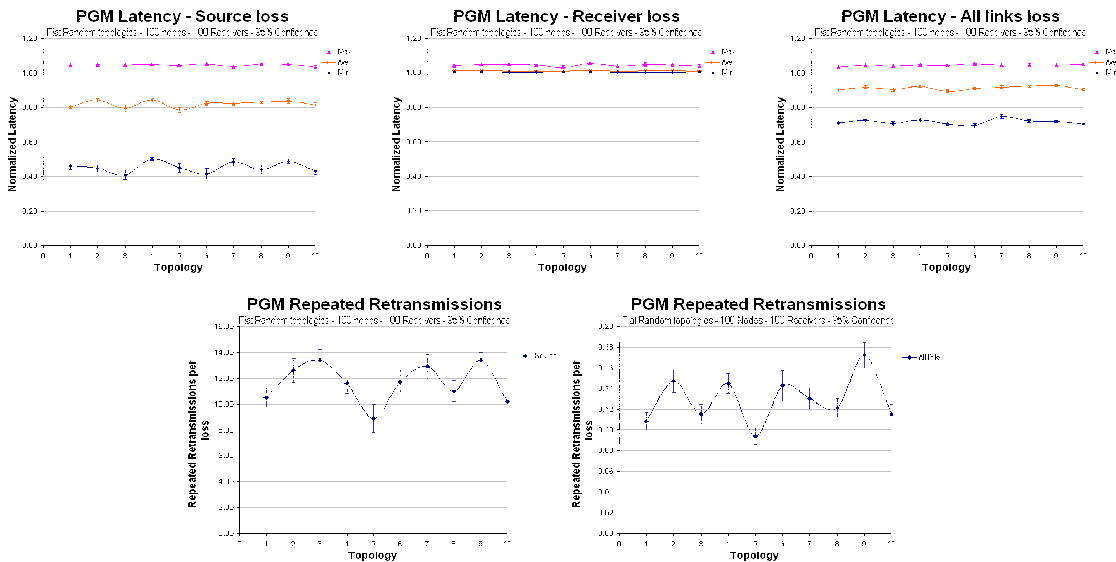


Figure 14: PGM with random topologies

cast latency. One might wonder why this experiment did not produce results similar to LMS, since in both cases recovery is done from the source; the reason is due to PGM repeated retransmissions, which result in an increase of the average recovery latency. In addition, recall that in PGM receivers observe a random back-off before sending NACKs, which results in an increase in the overall recovery latency. We have used a back-off value of zero in our experiments; with larger back-off values, recovery latency will increase but repeated retransmissions will diminish.

With loss at the receivers, the recovery latency is very close to 1, as expected. Note that with PGM recovery latency does not increase much beyond 1 because a retransmission always comes from the source. Thus, unlike LMS, PGM does not allow a receiver to send a retransmission which may result in a longer retransmission path. With loss at all links, recovery latency in PGM increases slightly (a trend also seen with LMS), to about 90% of unicast latency. This is about 30% more than what is seen with LMS. In summary, it appears that on average, allowing receivers to participate in recovery saves about 30 - 50% in recovery latency.

We also measured the PGM repeated retransmissions when loss occurs near the source. Loss that occurs near the source will create the maximum number of repeated retransmissions. Our results show that the

number of repeated retransmissions can be quite high, between 9 and 13 retransmissions for each lost packet. This experiment did not include any back-off at the source, because it was not part of the initial PGM specification. Since then, the PGM designers have added a heuristic to reduce this problem, where the source delays the sending of a retransmission to allow NACKs to establish state at the routers. The difficulty in designing such a heuristic is how to determine the appropriate amount of back-off to minimize repeated retransmissions while not unduly increasing recovery latency.

We also measured repeated retransmissions with loss distributed at all links. It is clear that the problem becomes much less pronounced in this experiment. It appears that the number of repeated retransmissions reduces by two orders of magnitude. Thus is hard to estimate the real-life effect of repeated retransmissions in PGM until we have a reasonably good loss model. What this experiment clearly demonstrates again is the sensitivity of multicast protocols to topology and loss location. By definition, no repeated retransmissions occur with loss at the receivers.

3.6. Discussion

In general, it appears that network support enables LMS and PGM to perform much better than SRM. Improvements can be seen in both recovery latency, exposure and duplicates. We believe that these performance advantages are important enough to warrant serious consideration for deployment; for example, lower latency may make multicast error control feasible for interactive real-time applications; the absence of exposure, is important for scalability. PGM and LMS have the additional advantage of freeing receivers from maintaining any topology-related state and performing any topology related operations. This significantly reduces application complexity, and may ease the development of multicast applications.

Comparing LMS and PGM, we note that while LMS is much simpler to implement and requires significantly less overhead than PGM, its performance is on par with PGM. LMS has significantly lower recovery latency, while trading very little in terms of exposure. PGM incurs per-lost-packet state at the routers which may be significant for large backbone routers. In contrast, LMS incurs only a small fixed state penalty per multicast group at the routers.

The performance results are summarized in Table 1. With LMS a receiver typically recovers from loss

Table 1: Performance summary

SCHEME	Normalized Latency (% of unicast latency)	Exposure/duplicates	Repeated retransmissions
LMS	30 - 60%	0.5%	none
SRM	>200%	4-6 duplicates per loss	none
PGM	80 - 100%	none	up to 10 - 13 per loss

in about 30-60% of its unicast latency to the source. PGM, which always recovers from the source, requires about 80-90% of the unicast latency. Thus it seems that the penalty PGM pays in terms of latency by not allowing local recovery is about 30-50%. SRM, due to its duplicate suppression mechanism, requires the longest time to recover, which is at least twice the unicast latency.

In terms of duplicates, SRM generates about 4-5 duplicate packets per loss. Without some form of local recovery, these duplicates will be sent to all receivers. LMS and PGM only allow one retransmission to reach a particular receiver. However, whereas PGM will never allow a retransmission to reach a receiver that has

not sent a NACK, LMS may allow unwanted retransmissions to reach receivers that do not need them. Our simulations show that the probability of a receiver receiving unwanted packets is not very high, even with static repliers: typically, under 0.5%. PGM, while preventing unwanted packets at the receivers, may force the source to send repeated retransmissions in response to a single packet loss. The number of these retransmissions can be high enough (about 10 - 13 per loss) to be of some concern; however, without a better understanding of loss characteristics on the network, it is hard to quantify the overhead of repeated retransmissions. What is clear though, is that PGM will have to incorporate some delay before sending retransmissions, which will increase its recovery latency beyond what we have reported here.

4. Processing Overhead

We have evaluated the processing overhead of our LMS implementation using the testbed shown in Figure 15. Even though our topology is very simple, it still allows the measurement of processing cycles required for all LMS forwarding operations. We measured the forwarding overhead at the LMS router; we did not measure the additional processing at the hosts because our changes there were minor. We used Pentium II class machines, connected together with a 155 Mbps ATM network. The multicast sender was on host marked SRC; hosts H1 and H2 are the receivers. Host H1 (shaded) was the replier. The measurements were taken using the processor cycle counter register in the Pentium processor. We measured the processing at the entire IP layer, from the moment a packet was received at IP until the packet was passed to the network interface. The machines we used were all 300MHz Pentium II class machines. The number of cycles was counted from when a packet entered the IP layer (at the beginning of function `ipintr`) until the packet exited the IP layer (right before `ip_output` calls the first network layer function).

We first verified the correct operation of LMS. After verifying that all LMS forwarding functions worked correctly, we set up our experiments to measure processing overhead.

Baseline Experiments

We ran two baseline experiments: in the first we sent about 6 million packets from SRC while only H1 was a member of the multicast group; in the second experiment we sent the same number of packets from SRC, but with both H1 and H2 being members. We measured the number of cycles spent at the router to forward packets in both experiments. These numbers provided us with a baseline estimate of how many cycles it takes to forward a regular multicast packets. The results are shown in Table 2.

LMS Experiments

We measured the processing overhead of the forwarding operations at the LMS router by running two experiments: one to measure the processing overhead to forward a request, and another to measure the overhead for a dmcast. In the first experiment, host H2 sent about 6 million requests to the replier, which the router received and forwarded to H1. In the second experiment, host H1 sent about 6 million dmcasts which were multicast on the interface leading to H2. The results of these experiments along with the baseline experiments, are summarized in Table 2.

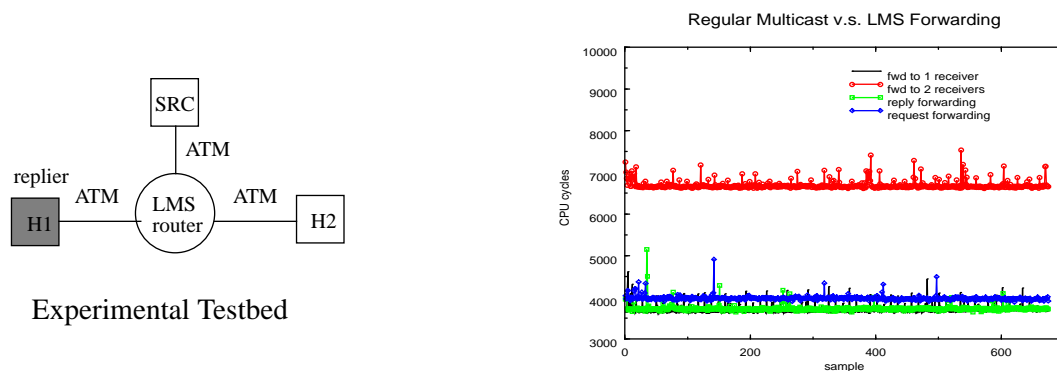
Table 2: Forwarding cost: Normal v.s. LMS processing (300MHz Pentium II)

Normal IP mcast forwarding to 1 receiver	Normal IP mcast forwarding to 2 receivers	Forwarding a LMS request	Forwarding a LMS dmcast
3702 cycles	6686 cycles	3979 cycles	3734 cycles

Table 2: Forwarding cost: Normal v.s. LMS processing (300MHz Pentium II)

Normal IP mcast forwarding to 1 receiver	Normal IP mcast forwarding to 2 receivers	Forwarding a LMS request	Forwarding a LMS dmcast
12.3 μ s	22.3 μ s	13.3 μ s	12.4 μ s

The Table shows the average number of cycles spent at the IP layer to process each packet in the four experiments we described. The Table also shows the average number of microseconds taken to process each packet, which we obtained simply by dividing the number of cycles with the processor speed. A more detailed breakdown of the results is shown in Figure 15.

**Figure 15: Experimental testbed and LMS forwarding cost**

As we can see from both the table and the plot, the cost of forwarding LMS packets is approximately the same as the cost of forwarding a single multicast packet. It appears that the forwarding cost of regular multicast packets increases almost linearly as the router has more member interfaces. The cost of LMS packets however, by design remains constant, regardless of how many member interfaces the router has.

The important result of this section is that the cost of forwarding LMS packets is on par with the cost of forwarding a regular multicast packet; moreover, it remains constant regardless of the router fan-out. This shows that LMS processing at the routers is *not* a bottleneck.

5. Overview of Related Work

There has been a significant amount of research on reliable multicast protocols. The early work has focused on distributed systems, providing primitives for constructing distributed applications, such as the ISIS system[30] and the V-kernel[31]. Other early work has focused on local area networks or broadcast links [32, 33, 34, 35]. A good survey of the early work can be found in [27]. Here we focus on recent work on reliable multicast that aims to provide scalability to very large groups.

The vast majority of recent reliable multicast protocols use receiver-reliable recovery, shown by Pingali, Towsley, and Kurose to be superior to sender-reliable recovery [11]. We divide reliable multicast schemes into *non-assisted* and *assisted schemes* based on whether they require router assistance or not. We begin with non-assisted schemes.

RMTP[10] is an example of a static hierarchical scheme. The source multicasts data to all receivers, but only the *Designated Receivers* (DRs) return acknowledgments. Losses in RMTP are recovered from DRs. Retransmissions are either unicast or multicast depending on how many requests were received. Although not implemented, RMTP was the first protocol to propose the use of *subcast*¹, a router service that allows a router to multicast a packet to all downstream links.

The Log-Based Receiver-reliable Multicast (LBRM) [7] is another example of a static hierarchical scheme, aimed at distributed interactive simulation (DIS) applications. LBRM uses a primary logging server and a static hierarchy of secondary logging servers which log all transmitted data. Data is multicast from the source to all logging servers and all receivers; however, only the primary logging server returns acknowledgments to the source. The receivers request lost data from the secondary logging servers; in turn, the secondary logging servers request any lost data from the primary logging server. Similar to RMTP, retransmissions in LBRM are either unicast or multicast, or multicast based on a threshold. Both RMTP and LBRM are based on a static hierarchy and thus require explicit set-up of DRs or logging servers before new regions can be added to the group.

The Tree-based Multicast Transport Protocol (TMTP) [14] is an example of a scheme that uses a dynamic hierarchy. In TMTP, every region has a Domain Manager (DM). When a DM joins a group, it searches for a parent using an expanding ring search. During the search, the new DM repeatedly broadcasts a “SEARCH_FOR_PARENT” request by increasing the time-to-live (TTL) value. When one or more DMs respond, the new DM selects the closest DM as its parent. Thus, the DMs form a dynamic hierarchical control tree. Each endpoint maintains the hop distance to its DM, and each DM maintains the hop distance to its farthest child. These values are used to set the TTL field on requests and replies to limit their scope. To further limit request implosion at the DMs, TMTP uses randomized backoff for requests, which, however, increases latency.

LGMP[36] is a hierarchical, subgroup-based protocol, where receivers take the responsibility to dynamically organize themselves into subgroups. Subgroups select a Group Controller to coordinate local retransmissions and process feedback messages. LGMP subgroups are self-organizing and self-adaptive according to the current network load and group membership. However, LGMP subgroup topology may not always be congruent with the underlying multicast tree. LGMP has been implemented and some of its testing was carried out on the MBONE.

TRAM[38] is another dynamic tree-based protocol designed to support bulk data transfer. TRAM uses TTL to form the receiver tree. The tree formation and maintenance algorithms borrow from other schemes like TMTP, but TRAM has a richer tree management framework, supporting member repair and monitoring, pruning of unsuitable members, and aggregation and propagation of protocol related information.

MFTP[37] is designed for reliable distribution of files to a large number of receivers. Data is transmitted in passes. After each pass, receivers unicast NACKs back to the sender using random back-off delay to avoid implosion. The sender collects all NACKs and transmits all missing packets in the next pass. The process repeats until all receivers receive the data and no NACKs are sent. It is clear that MFTP trades latency for reliability, a trade-off which is acceptable for file transfer, but may not be acceptable for other applications.

Static hierarchical schemes like RMTP and LBRM do not adapt to rapid membership changes or changes in topology. Dynamic hierarchical schemes like TMTP, LGMP, and TRAM rely on approximate methods (e.g., expanding ring search) to discover parents and send replies. The use of expanding ring

1. Term coined by Adam Costello.

search for parent selection can lead to other problems, due to lack of congruency between the recovery tree and the underlying topology. Other schemes like MFTP are mostly suited for bulk data transfer.

5.1. Schemes Using Forward Error Correction (FEC)

FEC is attractive in a multicast environments with a high degree of uncorrelated loss, because such losses can be repaired efficiently. FEC typically increases the bandwidth required to transmit data, depending on the encoding method used. Recently, techniques have been proposed that reduce this overhead and increase the effectiveness of FEC[40, 41, 39]. We chose to investigate retransmission in our work because it offers very low cost in terms of bandwidth and does not require encoding/decoding of data at the ends. Some of the techniques we have devised in our work can be used with FEC solutions, for example in sending scoped parity packets.

5.2. Router-Assisted Schemes

Recently, there have been several proposals to use network assistance for reliable multicast, which we describe below. Most of these postdate our work.

In Addressable Internet Multicast (AIM)[8], the authors propose to extend Internet routing by defining a rich set of services. These services require routers to assign per-multicast group labels to all routers participating in that group. There are three types of labels: positional, distance, and stream labels. Positional labels are used to route messages to individual members of the group. Distance labels are used to locate near-by members. Stream labels are used to subscribe to traffic generated by a subset of sources. AIM defines new routing mechanisms based on the presence of these labels. In the Reliable Multicast Architecture (RMA), members requiring a retransmission ask their local router to send a request using a positional reachcast towards the source. A reachcast eventually reaches members that have the requested data, which respond by sending a retransmission via positional routing. The proposed labeling scheme has less overhead when used in shared trees. If used in source-based trees, each source tree requires its own labels. The overhead of distributing the labels after a membership change can be high if groups are highly dynamic: whenever a new branch is added to the multicast tree, all the routers below the new branch may have to change their labels.

Search Party[6] builds on our work in LMS by using randomness to enhance robustness. In Search Party, requests are not routed deterministically, as in LMS, but randomly using a new mechanism called “randomcast”. This mechanism is used by routers to randomly route requests to either the parent or to one of the children. Search Party trades efficiency (in terms of increased latency and duplicates) for better robustness.

OTERS[42], uses a modified version of the mtrace[29] utility to construct a recovery tree that is congruent with the underlying multicast tree. OTERS builds the tree by incrementally identifying subroots in the multicast routing tree using back-tracing. For each subroot, OTERS selects a Designated Receiver (DR) which acts as the parent. OTERS solves the problem of maintaining congruency (in other words, ensuring that the recovery tree mirrors the underlying multicast tree), but receivers are still exposed to topology and have to keep track of changes in the structure of the underlying multicast group. In addition, the overhead of using mtrace probes may be high in highly dynamic groups.

Tracer[28] is similar to OTERS in that it also uses the mtrace utility to allow each receiver to discover its path to the source. Once the path is discovered, receivers advertise their paths to near-by receivers using expanding ring search. Once receivers discover nearby receivers, they use the data from tracing and their loss rate to select parents. Tracer can be used as a facility to create congruent trees for other tree-based pro-

ocols, such as RMTP. As with OTERS, Tracer exposes receivers to the underlying topology of the group and incurs overhead due to mtrace probes.

6. Conclusions

Multicast deployment has been hampered by many factors, including the lack of multicast applications that truly scale to a large number of receivers. In this paper we presented LMS (Light-weight Multicast Services), a set of forwarding services that enhance the current multicast service model with functionality that allows the implementation of highly scalable multicast applications. We have shown that LMS is simple to implement, does not violate the end-to-end principle and incurs performance overhead comparable to normal multicast. We have covered in depth the implementation of a reliable multicast scheme using LMS. We have shown, however, how other multicast applications can leverage off the virtual hierarchy built by LMS. Such a hierarchy offers excellent scaling characteristics, but is very hard to build and maintain without router assistance. The hierarchies built by LMS are also very light-weight and flexible because their construction is receiver-driven. We have demonstrated through simulation that with LMS the performance of error recovery improves significantly over non-assisted schemes: implosion is eliminated, exposure is kept at negligible levels, and recovery latency is nearly optimal. In addition to improving performance, LMS greatly simplifies receivers by freeing them from the burden of topology discovery.

An important contribution of our work is the observation that forwarding and error control are two clearly separable components, and great benefits can be realized by decoupling and placing each one where it is more beneficial. We believe that the forwarding component belongs to the routers (after all this is what routers do best), and the actual error control component (which is an end-to-end operation) belongs to the receivers. This separation is very clean, i.e., it does not violate any layering principles like the end-to-end argument - the routers do not see any transport layer information and the endpoints know nothing about topology. This separation has provided us with a vantage point to address other important, yet difficult multicast problems like congestion control, ACK-based reliable multicast, and topology related grouping.

Acknowledgments

Many thanks to Sherlia Shi for writing the ns code for PGM and for the long discussions.

REFERENCES

- [1] Tanenbaum, A., Computer Networks, Prentice Hall, 1988.
- [2] Comer, D., Internetworking with TCP/IP, Prentice Hall, 1991.
- [3] Floyd, S., Jacobson, V., McCanne, S., Liu, C., Zhang, L., "A Reliable Multicast Framework for Light-Weight Sessions and Application Framing," Proc. of ACM Sigcomm '95, pp. 342-356, Cambridge MA 1995.
- [4] Lin, J., Paul, S., "RMTP: A Reliable Multicast Transport Protocol", Infocom '96, pp.1414-1424, March 1996.
- [5] Deering, S., "Host Extensions for IP Multicasting," RFC 1112, January 1989.
- [6] Costello, A., McCanne, S., "Search Party: Using Randomcast for Reliable Multicast with Local Recovery", Proceedings of INFOCOM'99, March 21, 1999, New York, NY.
- [7] Holbrook, H., Singhal, S., Cheriton, D., "Log-Based Receiver-Reliable Multicast for Distributed Interactive Simulation," Proceedings of ACM Sigcomm'95, Vol. 25, No. 4, pp. 328-341, October 1995.
- [8] Levine, B., Garcia-Luna-Aceves, J.J., "Improving Internet Multicast with Routing Labels", Proc. of IEEE ICNP, Atlanta, GA, Oct. 1997, <http://www.cse.ucsc.edu/research/ccrg/publications.html>.
- [9] Papadopoulos, C., Parulkar, G., Varghese, G., "An Error Control Scheme for Large-Scale Multicast Applications", Proc. of IEEE INFOCOM'98, San Francisco, CA pp.1188-1196, March 1998.

- [10] Paul, S., Sabnani, K., Buskens, R., Muhammad, S., Lin, J., Bhattacharyya, S., "RMTP: A Reliable Multicast Transport Protocol for High-Speed Networks," Proceedings of the Tenth Annual IEEE Workshop on Computer Communications, September 1995.
- [11] Pingali, S., Towsley, D., Kurose J., "A Comparison of Sender-initiated and Receiver-initiated Reliable Multicast Protocols," JSAC, April 1998.
- [12] Saltzer, J.H., Reed, D.P., Clark, D.D., "End-to-End Arguments in System Design," ACM Transactions on Computer Systems, Vol. 2, No. 4, November 1984. pp 277, 288.
- [13] Yajnik, M., Kurose, J., Towsley, D., "Packet Loss Correlation in the MBONE Multicast Network: Experimental Measurements and Markov Chain Models," IEEE Global Internet Conference '96.
- [14] Yavatkar, R., Griffioen, J., Sudan, M., "A Reliable Dissemination Protocol for Interactive Collaborative Applications," Multimedia'95.
- [15] Stevens, W., Thomas, M., "draft-stevens-advanced-api-03.txt", work in progress.
- [16] Gilligan, R., Nordmark, E., "Transition Mechanisms for IPv6 Hosts and Routers", RFC 1933, April 1996.
- [17] Katz, D., IP Router Alert Option, Request for comments, RFC 2113.
- [18] Partridge, C., Mendez, T., Milliken, W., "Host anycasting service," RFC 1546, November 1993.
- [19] Mittra, S., "Iolus: A Framework for Scalable Secure Multicasting," Proc. of the ACM Sigcomm '97, Cannes, France, September 1997.
- [20] Speakman, T., Farinacci, D., Lin, S., Tweedly, A., "Pragmatic General Multicast (PGM) Transport Protocol Specification", draft-speakman-pgm-spec-03.txt, work in progress, June 1999.
- [21] UCB/LBNL/VINT Network Simulator - ns (version 2), Software on line, <http://www-mash.cs.berkeley.edu/ns/>.
- [22] Zegura, E., Calvert, K., and Bhattacharjee, S., "How to Model an Internetwork." Proceedings of IEEE Infocom'96, San Francisco, CA.
- [23] Handley, M., An Examination of MBone Performance USC/ISI Research Report: ISI/RR-97-450, January 1997.
- [24] The Reliable Multicast Research Group, <http://www.east.isi.edu/rm/>.
- [25] Internet Protocol, Version 6 (IPv6) Specification <draft-ietf-ipngwg-ipv6-spec-v2-02.txt> (replaces RFC 1883), December 1998, work in progress.
- [26] 6Bone, <http://www.6bone.net/>.
- [27] B.N. Levine, David Lavo, and J.J. Garcia-Luna-Aceves, "The Case for Concurrent Reliable Multicasting Using Shared Ack Trees," Proc. ACM Multimedia 1996 Boston, MA, November 18--22, 1996.
- [28] B.N. Levine, S. Paul, and J.J. Garcia-Luna-Aceves, "Organizing Multicast Receivers Deterministically According to Packet-Loss Correlation," Proc. Sixth ACM International Multimedia Conference (ACM Multimedia 98), Bristol, UK, September 1998.
- [29] Fenner, W., Casner, S., "A traceroute facility for IP Multicast" Internet Draft draft-ietf-idmr-traceroute-ipm-02.txt, work in progress, 1997.
- [30] Birman, K., Joseph, T., "Reliable Communication in the presence of failures:", ACM Transactions on Computer Systems, 5(1):47-76, February 1987.
- [31] Cheriton, D., Zwaenepoel, W., "Distributed Process Groups in the V kernel", ACM Transactions on Computer Systems, 3(2):77-107, May 1985.
- [32] Chang, J., Maxemchuck, N., "Reliable Broadcast Protocols", ACM Transactions on Computer Systems, 2(3):251-273, August 1984.
- [33] Kaashoek, M., Tanenbaum, A., Humel, S., Bal, H., "An Efficient Reliable Broadcast Protocol", ACM Operating Systems Review, 23(4), October 1989.
- [34] Laurence C. N. Tseung, "Guaranteed, Reliable, Secure, Broadcast Networks", IEEE Network magazine, Nov. 1989, pp. 33-37.
- [35] Crowcroft, J., Paliwoda, K., "A Multicast Transport Protocol", Proc. of ACM Sigcomm'88, pp. 247-256, August 1988.
- [36] Hofmann, M., Home page of the Local Group Concept (IGC), <http://www.telematik.informatik.uni-karlsruhe.de/~hofmann/LocalGroups.html>.
- [37] Miller, K., Robertson, K., Tweedly, A., White, M., "StarBurst Multicast Transfer Protocol (MFTP) Specification", Internet Draft, draft-miller-mftp-spec-02.txt, work in progress, January 1997.

- [38] Chiu, D., Hurst, S., Kadansky, M., Wesley, J., “TRAM: A Tree-based Reliable Multicast protocol”, Sun Technical Report SML TR-98-66, Sun Microsystems, July 1998.
- [39] Vicisano, L., Crowcroft, J., “One to Many Reliable Bulk Data Transfer on the MBONE”, Third International Workshop on High Performance Protocol Architectures HIPPARCH '97, Sweden, June 1997.
- [40] Nonnenmacher, J., Biersak, E., Towsley, D., “Parity-based loss recovery for Reliable Multicast Transmission”, ACM/IEEE Transactions on Networking, 1998.
- [41] Rubenstein, D., Kurose, J, Towsley, D., “Real-Time Reliable Multicast Using Proactive Forward Error Corection”, NOSSDAV'98.
- [42] D. Li and D. R. Cheriton. OTERS (On-Tree Efficient Recovery using Subcasting): A Reliable Multicast Protocol, Proceedings of 6th IEEE International Conference on Network Protocols (ICNP'98). October 1998, Austin, Texas, pp 237-245.
- [43] Liu, C.-G., Estrin, D., Shenker, S., and Zhang, L., Local Error Recovery in SRM: Comparison of Two Approaches, USC Technical Report 97-648, January 1997.
- [44] Holbrook, H., Cheriton, D., “IP Multicast Channels: EXPRESS Support for Large-Scale Single-Source Applications” Proc. of Sigcomm'99, Boston, MA, August 1999.
- [45] Perlman, R., Lee, C-Y., Ballardie, A., Crowcroft, J., Wang, Z., Maufer, T., Diot, C., Thoo, J., Green, M., “A Design for Simple, Low-Overhead Multicast,” Internet Draft, draft-perlman-simple-multicast-02.txt, February 1999, work in progress.
- [46] “The State of the Internet: Roundtable 4.0,” IEEE Specrum, vol.35, No. 10, October 1998.
- [47] Diot, C., Levine, B., Lyles, B., Kassem, H., Balensiefen, D., “deployment Issues for the IP Multicast Service and Architecture,” <http://>
- [48] Wheten, B., Vicisano, L., Kernmode, R., Handley, M., Floyd, S., “Reliable multicast transport building blosks for one-to-many bulk data transfer,” Internet draft, draft-ietf-rmt-buildingblocks-01.txt, work in progress.
- [49] Papadopoulos, C., “Error control for continuous media and large scale multicast applications,” PhD thesis, Washington University, St. Louis MO, August 1999.

A. APPENDIX: Protocol Details

In this section we describe some of the details of LMS. We first identify some problems and discuss their solutions; then we proceed to possible enhancements.

A.1. Preventing Duplicate Retransmissions

Multicast error recovery protocols (including those using LMS) that allow receivers to send retransmissions, face an ambiguity problem when requests arrive at receivers asking for data just recovered through retransmission. We will call these requests *late requests*. The ambiguity problem is depicted in Figure 16. Assume that the links between R3-R2 and R4-R2 are links with long propagation delay, as shown on the left. Now suppose a packet is lost between R1 and R2. A request from E2 will reach E1 which will DMCAST the data to the subtree. Now suppose that the requests from R3 and R4 reach E2 after the reply; obviously E2 should ignore these requests. However, if we look at the right side of the figure, late requests may be legitimate if the retransmission happened to be lost again on its way to R3.

To overcome this problem, we propose that repliers do not serve late requests, unless they receive a second request. An optimization is for receivers to mark their first request, which can always be safely ignored by repliers. This is a conservative approach, but one that eliminates ambiguity. If some duplicates are acceptable, repliers can serve requests immediately (except those marked as first). Another possible solution is to introduce an “ignore” period at repliers, where repliers ignore requests for some time after receiving a retransmission. We expect that different applications will employ different methods of dealing with late requests.

directly connected to the root as in source-based schemes, the source is connected by a unicast path to the root.

The above modification works well for unidirectional shared trees but not for bidirectional trees like CBT, because CBT routers do not maintain state about the direction of a source. Thus, LMS in its current state does not work with bidirectional shared trees, unless some per source information is added to the router state. Note, however, that in applications that require source filtering this information may be available even in bidirectional shared trees.

A.4. Replier Failure

The failure of a replier may disrupt recovery until the failed replier is detected. Note, however, that replier failure becomes a problem only if either the requestor and replier closest to the loss fail. Thus, recovery will only be affected if the break in the replier continuity coincides with the location of loss. Failed repliers are typically detected by routers through soft state. However, if detection via soft state is too slow, we propose the following failure recovery mechanism, depicted in Figure 18:

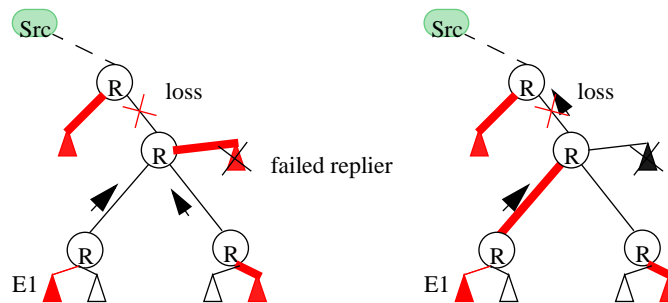


Figure 18: Detection of replier failure

- Requestors set a timer and repeat requests if the timer expires up to N times.
- After N attempts the requestor declares replier failure. The requestor then, sends a message to the router at the turning point, alerting the router of replier failure.
- The router switches repliers and forwards the request to the new replier.

While detection of a failed replier is taking place, the replier closest to the loss uses *heartbeat DMCASTs* to notify downstream receivers that failure recovery is in progress. A heartbeat DMCAST is special in the sense that it is forwarded on *all* downstream links, including the replier link. In the figure, E1 takes the responsibility to perform the necessary actions to restore the replier continuity. E1 knows that it is the closest to the loss and should initiate the replier recovery process because it receives no heartbeat messages, except its own. Receivers closer to the loss start their heartbeat earlier. All remaining receivers who receive a heartbeat, defer the repair to E1. These receivers can in turn monitor E1 through its heartbeat. E1 seizes its heartbeat after a retransmission is received.

A.5. Selecting Repliers in a LAN

For simplicity, the previous sections had assumed that there was only one receiver at each link. This is obviously not true when links terminate in LANs. In such cases, receivers on a LAN use a simple election mechanism to elect a replier and therefore make a LAN appear as having a single receiver. The details of how the election is done as well as other issues that arise with LANs are covered in [49].

A.6. Routers with a Large Fan-out

If a router has a large fan-out for a specific group, the router's replier may receive a large number of requests from downstream repliers. To avoid this problem, the router may partition its links into smaller groups and select a replier for every group, as shown in the example in Figure 19. In this example, requests

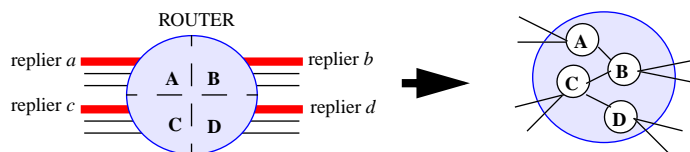


Figure 19: dealing with routers with large number of links

from links in group *D* go to replier *d*, but requests from replier *d* go to replier *c*, requests from replier *c* go to replier *b* and so on. Requests from replier *a* are forwarded upstream. By partitioning links this way, the maximum number of requests a replier can receive is significantly reduced.

A.7. Proxy Directed Multicast

In the previous examples we assumed that the first replier above the loss has the data and services the retransmission. This however, may not always be true. For example, since we employ a NACK-based protocol, the request may arrive after the buffers at the replier have been purged; or perhaps for security reasons, the retransmissions are only allowed to come from the original source or some other trusted member. Even in these cases, the DMCAST service can be used as before with only a small added overhead. We call this proxy directed multicast.

It is important to note that once a request passes the turning point it contains enough information to uniquely identify the subtree that requires the retransmission. Thus, if a replier receives a request that it cannot service, the replier can forward the request to another member (possibly using LMS again). Once the request arrives at an appropriate replier, a DMCAST can be sent to the original turning point, thus reaching the correct subtree. In order to preserve the original turning point, routers forwarding a request to a replier first check if the turning point field is empty. If it is not empty, the existing information is left untouched.

A.8. Pathological Topologies

Some topologies may create pathological situations that may reduce the efficiency of LMS. In this section, we describe two such cases and propose possible solutions to overcome the problem. It is not clear how likely these topologies are to occur in real-life. The topologies generated for our simulation experiments did not exhibit any such pathologies.

A.8.1. Long and Skinny Branches

These topologies may cause increased recovery latency in LMS. An example is shown in Figure 20. These are essentially topologies composed entirely of long, skinny branches with no repliers in the middle. Latency may be increased in such topologies because router *R* has no choice but to select one of the long branches as the replier link. Thus, requests from a non-replier branch travel all the way back (possibly near the source) only to be diverted down another long branch to the replier. The same applies to retransmissions: they must follow the long path through the turning point.

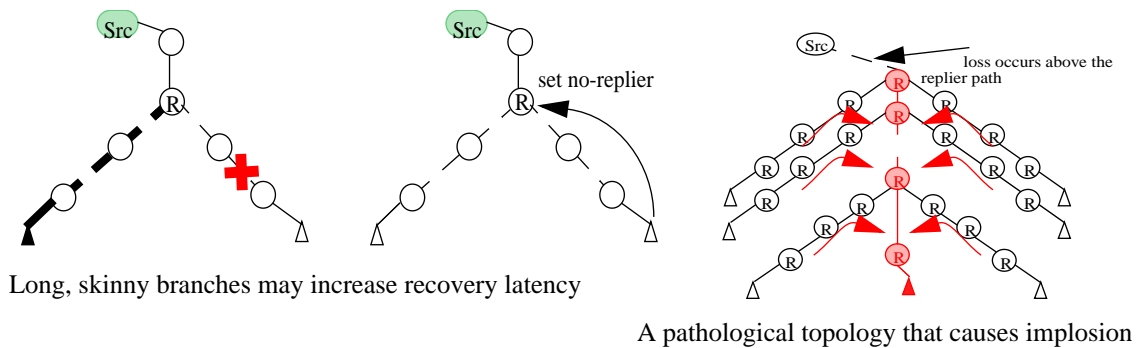


Figure 20: Some pathological topologies

A possible solution to this problem is the following: receivers that experience long recovery latency (compared to their RTT to the source) due to such topologies, advise the turning point router to release its repplier and propagate requests from all downstream branches upstream, in the hope that another repplier may be located closer to the turning point. The router may or may not follow this advice: if the number of downstream links is small thus not risking implosion at the upstream repplier, then this is a viable solution. Otherwise, receivers will have to deal with the increased latency. Note that if the router follows this advice, it should continue to insert the turning point information to requests so that dmcasts will be efficient. A disadvantage of this approach is that if loss happens upstream of the turning point, then one DMCAST per downstream link will be required at router R.

A.8.2. Topologies that cause Request Implosion

A pathological topology that may cause request implosion is shown in Figure 20. Recall that a router forwards at most one request on its upstream link. Thus, the maximum number of requests a repplier can receive is typically bounded by the number of downstream links at the turning point. However, it is conceivable that in such pathological cases where many routers have selected the same repplier, the repplier may receive a potentially large number of requests. In the figure, a large number of neighboring routers (shaded) have selected the same repplier (also shaded), forming a long repplier path. Every request reaching a router on the repplier path is now forwarded to the same repplier, making the number of requests at the repplier proportional to the sum of the downstream links of all the routers on the repplier path.

The problem can be solved by modifying the repplier cost to take into consideration the sum of the children of all routers along the repplier path. At each hop, a router modifies the cost to reflect the number of children (minus the repplier link). Thus as it moves upstream, the cost becomes high and the next upstream router is forced to select a different repplier link, thus shortening the repplier path.

A.9. Security Issues

Current multicast security schemes employ encryption techniques to ensure security in multicast groups [19]. With encryption techniques, the receivers are given keys, which they use to decrypt packets sent by the sender. There are two possible problems that may arise with secure applications in LMS. These are the following:

- Data integrity: how can the requestors ensure that retransmissions sent by repliers contain the same data originally sent by the source?
- Unresponsive repliers: how can requestors ensure that repliers will respond to their requests?

We address the first problem by requiring repliers in secure applications to buffer and retransmit packets in their encrypted state, i.e., as they were received from the sender. In addition, LMS routers may enforce the rule that DMCASTs contain the address of the replier in an IP option to alert receivers. The second problem can be addressed by treating unresponsiveness as failure, and switching repliers as described earlier.

A.10. Incremental Deployment

Incremental deployment is an important issue in the deployment of any new scheme requiring changes at routers. The Internet has become so big, that a clear incremental deployment plan is essential. The easiest way to deploy LMS is to follow the same plan as IPv6[25, 16] deployment. IPv6 is currently running on an experimental overlays (e.g., 6-Bone [26]). Another option is to deploy LMS following a similar approach as the one proposed for the incremental deployment of PGM [20]. PGM uses *Source Path Messages* (SPMs) to create an overlay of PGM aware routers. PGM signalling then occurs only between PGM routers. The SPM approach works for LMS, as depicted in Figure 21 and described below.

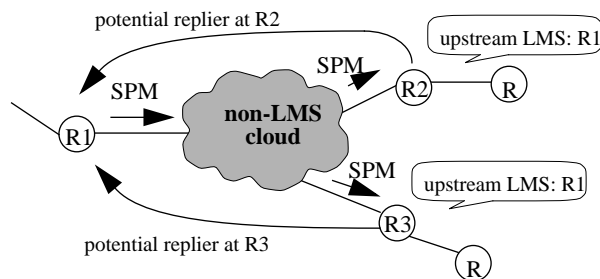


Figure 21: LMS Incremental Deployment

Periodically, the source multicasts SPMs to the group. These also carry the IP Router Alert option, so that all routers examine them. Non-LMS routers simply ignore the option and forward packets in the normal fashion. When an LMS router receives an SPM, it notes the address of the sender. If the sender is not its upstream neighbor, then it notes the address of the upstream LMS element, overwrites its own address in the SPM and forwards it. This process will eventually create pointers to LMS elements along the reverse path. Unlike PGM, LMS requires an additional step. LMS needs to send messages to repliers, so a router also needs to know the forward path to the replier. The forward path is established by replier updates traveling upstream using the same mechanism as SPMs.

Incremental deployment has an effect on exposure. For example, a retransmission initiated at the turning point will reach receivers in intermediate non-LMS clouds. If loss occurred upstream the clouds, the retransmission will be useful to all receivers; however, if loss occurred downstream, receivers in the clouds will receive a duplicate.

A.11. Other LMS Applications

One of the important advantages of LMS is that its mechanisms are general enough to be used for other purposes besides error recovery. For example, the services used for implosion control can be used by many other applications where receivers need to implement a *scalable collect* or *voting* service. Such a service enables a large number of receivers to efficiently convey some value to the sender without risking implosion. The fine-grain multicast capability provided by the DMCAST service can be used for targeting specific parts

of the multicast tree, for example to announce the presence of a server in that region. We expand on these applications next.

A.11.1. Simple ANYCAST

ANYCAST [18] is a service used when a client wishes to discover one server (typically the nearest one) out of a group of servers providing a service the client is interested in. Examples are replicated file systems, or the DNS service. LMS can be used to implement a simple ANYCAST service by grouping servers in a well-known multicast group. Servers use LMS to notify routers that they want to be repliers, choosing perhaps distance as the replier metric. Clients searching for servers send requests to the multicast group, which are directed to the nearest replier (as with retransmission requests) thus establishing contact with the server. These steps are shown in Figure 22.

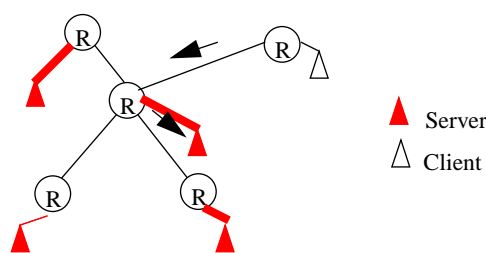


Figure 22: LMS implements a simple ANYCAST service

One small change required in the replier selection method to implement ANYCAST. Servers tell the routers to advertise the existence of a replier (server) in all links rather than just the upstream link as before, which ensures that routers find the nearest server in any direction.

A.11.2. Positive Acknowledgment-Based Reliable Multicast

Recall from the previous section, that NACK-based protocols cannot guarantee 100% reliability due to the lack of positive ACKs. To achieve complete reliability, one should either employ a positive ACK-based protocol, or supplement NACKs with periodic synchronization via ACKs. The mechanisms for both approaches are similar; the main difference is the frequency of synchronization, which happens on every packet (or window) in an ACK-based protocol, instead of at a larger interval with a NACK/ACK combination.

LMS can be used to implement either a pure ACK-based protocol, or NACK/ACK protocol. In either case, repliers take the responsibility to collect ACKs from downstream receivers, fuse them, and send a cumulative ACK upstream. Thus the sender finally receives a single ACK containing the highest consecutive sequence number from all receivers.

LMS can help simplify the implementation of such protocols. For example, additional mechanisms that would be needed to assign children to parents (as done in existing protocols using static hierarchy [10]), have been eliminated; tree congruency, a nagging problem with other protocols, is now automatic; a joining receiver searching for a parent can easily find one upstream by sending a request; switching to a new parent after the current leaves the group is again easy: it simply requires that downstream receivers send a probe message searching for a new parent; finally, the DMCAST service is still available to send retransmissions efficiently.