Light-Weight Multicast Services (LMS): A Router-Assisted Scheme for Reliable Multicast

Christos Papadopoulos, *Member, IEEE*, Guru Parulkar, and George Varghese, *Member, IEEE*

***Abstract*—Building on the success of unicast IP, IP Multicast adopted a simple, open, best-effort delivery model with many-to-many semantics. Despite several years of effort, a general, scalable and reliable end-to-end transport protocol analogous to TCP has proven elusive. Proposed solutions are either inflexible, or incur high control overhead.**

We present Lightweight Multicast Services (LMS), which enhance the IP Multicast model with simple forwarding services to facilitate scalable and efficient (compared to pure end-to-end) solutions to problems such as reliable multicast. In LMS, routers tag and steer control packets to preselected endpoints and perform fine-grain multicast to guide responses to a subset of the group without transport-level processing.

LMS divides error control into transport and forwarding components, which allows the former to remain at the end-points while the latter is pushed to the routers, where it can be implemented very efficiently. The division is clean, resulting in significant gains in performance and scalability, while reducing application complexity. LMS reaches beyond reliable multicast to applications such as scalable collect, any-cast, and in general, any application that can benefit from a hierarchy congruent with the underlying topology.

***Index Terms*—Error control, multicast, reliable multicast.**

I. Introduction

AT THE CORE of the Internet architecture lies the simplicity and elegance of IP and its design principles [11]. Internet architects realized early on that by foregoing the wire-like robustness of traditional communications networks (such as the telephone network) and pushing the intelligence to the edges, a network can be built on a much simpler, cheaper, and highly scalable infrastructure.

The resulting *best-effort* service model has proven highly flexible. However, the interaction of store-and-forward packet forwarding, finite buffers, and bursty sources occasionally leads to congestion and loss. Applications requiring better than best-effort reliability must counteract loss with *error control*, the component of a communication protocol responsible for reliability [1], [2].

Manuscript received July 11, 2000; approved by IEEE/ACM TRANSACTIONS ON NETWORKING Editor S. Paul. An earlier version of this paper was presented at the IEEE INFOCOM, San Francisco, CA, 1998.

C. Papadopoulos is with the Department of Computer Science, University of Southern California, Los Angeles, CA 90089-0781 USA (e-mail: christos@isi.edu).

G. Parulkar is with Computer and Network Systems, National Science Foundation, Arlington, VA 22230 USA (e-mail: gparulka@nsf.gov).

G. Varghese is with the Department of Computer Science and Engineering, University of California at San Diego, La Jolla, CA 92040-0114 USA (e-mail: varghese@cs.ucsd.edu).

Digital Object Identifier 10.1109/TNET.2004.828938

IP Multicast [5] debuted in the late 1980s and was hailed as a natural extension of the unicast model. Multicast is a powerful service because it allows a single source to reach a virtually unlimited number of receivers in a very efficient and scalable manner. Multicast is well suited for applications such as streaming media, distance learning, Internet radio and television, distributed interactive simulation, file transfer, software updates, and much more. Continuing the architectural tradition of unicast, IP Multicast adopted a simple, best-effort, anonymous, broadcast-like service, often compared to a radio dial-tone: anyone may tune in and anyone may transmit. Thus, similar to unicast, IP Multicast provides a general service on top of which richer services can be built.

Despite the vigorous promotion of multicast by both the research and industry communities, the Internet Service Providers (ISPs) and the users have not yet embraced the service. Many reasons have been cited [38], [39], which include difficulties with inter-domain routing, peering relationships, address allocation, limited address space, security, billing, and the lack of a general scalable and reliable transport service analogous to TCP.

This paper addresses the last issue. We present Lightweight Multicast Services (LMS), an extension to IP Multicast, on top of which a general and scalable reliable multicast transport service can be constructed. LMS extends IP Multicast with a set of simple and lightweight services that enhance router forwarding to enable highly scalable, *network-assisted* solutions to reliable multicast. LMS cleanly separates the transport and forwarding components of error control, keeps the former at the endpoints thus avoiding layer violations, and pushes the latter to the routers where it can be implemented most efficiently.

Unicast error control mechanisms are not suitable for large-scale multicast due to the many-to-many nature of IP Multicast. Losses in multicast typically affect part of the multicast tree and attempting to recover localized loss leads to the following problems.

- **Implosion** occurs when the loss of a packet triggers redundant messages (requests and/or retransmissions). In large multicast groups, such messages may swamp the group and the network.
- **Exposure** occurs when recovery-related messages reach receivers that have not experienced loss. Exposure wastes both network and end-system resources.
- **Recovery latency**, defined as the latency experienced by a member from the instant a loss is detected until a reply is received, impacts buffering require ments and application utility.
- **Adaptability**: frequent changes in group membership and network conditions impact the efficiency of error

recovery (in terms of loss of service, redundant messages, additional processing, and/or latency), particularly when tenuous assumptions are made about receiver population and/or topology.

Current end-to-end solutions solve some, but not all, of the above problems. SRM [3] solves implosion, but at the expense of increased latency and exposure. RMTP [4] solves implosion, exposure, and latency, but at the expense of adaptability. TMTP [13] adapts to dynamic group membership and network conditions, but uses complex heuristics.

Briefly, LMS works as follows: as receivers join a multicast tree, they are organized by the routers in a hierarchy with each router dynamically selecting a parent. Upon detecting loss, all requests from children are steered toward the parent, while the request from the parent is forwarded upstream, ensuring that only one request escapes each subtree. Before funneling requests to the parent, a router inserts the address of both the incoming and outgoing interfaces in passing requests. We call such a router the *turning point* of the request, which identifies the root of the subtree that originated the request. The process ensures that a request will find a receiver that has the requested data, or reach the sender. In either case, a retransmission is unicast to the turning point router, which in turn multicasts it to the affected subtree.

Note how LMS addresses all the previous problems. Implosion and exposure are addressed by constructing a hierarchy, which localizes recovery between parents and children. The hierarchy adapts quickly to both group membership and routing changes since routers ensure that it always tracks the multicast routing tree. Recovery latency is minimized because with LMS the endpoints closest to the loss are involved and recovery messages are sent immediately. Finally, the router-maintained hierarchy eliminates all topology-related state from the receivers, such as timers, hop counts, parent/child relations, etc., and most associated signalling overhead.

This paper[1] is structured as follows. In Section II, we describe the basic operation of LMS. Section III presents additional protocol details. Section IV presents simulation results, including comparison between LMS and two other related prominent schemes, namely SRM [3] and PGM [16]. Section V presents measurements of our LMS implementation in the kernel of NetBSD Unix. Section VI discusses related work and Section VII concludes the paper.

II. Lightweight Multicast Services

LMS is a small set of forwarding services which enhance IP multicast to allow routers to automatically build an application-driven hierarchy and exchange packets between the different levels of the hierarchy. In this section, we first discuss why a hierarchy enables very efficient multicast error recovery. Then, we describe why this is difficult under the current multicast model. Finally, we show how LMS addresses this problem.

In Fig. 1, we observe a subset of receivers that have just experienced loss after a packet was dropped on link L. Assuming a nearby receiver has the data and is willing to retransmit, we call this receiver a *replier* and the one sending a request a *requestor*. Recovery is initiated by the requestor sending a NACK directly to the replier, followed by a multicast by the replier at link L. Recovery latency is minimized if the requestor and the replier are closest to the loss. We refer to this recovery process as *near-best*[2] because it eliminates implosion and exposure and minimizes recovery latency.

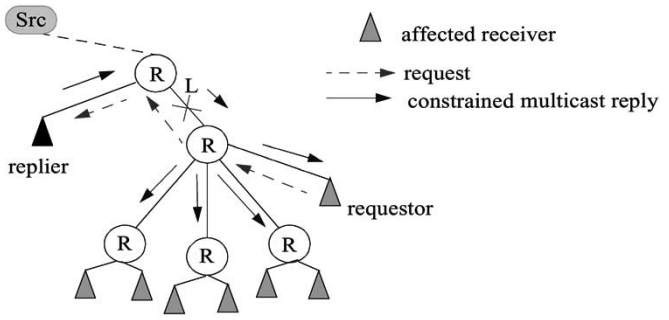


Fig. 1. Idealized recovery scenario.

This scenario cannot be realized with the current multicast model due to the lack of support to build such a hierarchy, find the closest receivers above and below a loss and target replies to a particular subtree.

A. LMS Concept

A router-based hierarchical solution, that is, one where routers temporarily buffer data and send retransmissions in response to NACKs, is architecturally incompatible with IP because it requires transport level processing at routers; yet it is attractive because it is conceptually simple and elegant. LMS reconciles this incompatibility by making the following key observation: a router-based solution is desirable not because it harnesses the router's processing power, but because it exploits the router's *location*. A natural question then is, would it be possible to move transport-level processing away from the routers (thus breaking the architectural incompatibility) while maintaining the location advantage?

The answer is yes. LMS achieves this goal by first decomposing error recovery into a *transport* and *forwarding* component and then moving the transport component from the router to a *surrogate*, leaving behind at the routers a set of forwarding services to steer packets between the router and the surrogate. By assigning surrogate responsibilities to end-points LMS pushes transport level recovery operations to the endpoints, thus avoiding layer violations. The cost is slightly increased latency compared to a router-based approach. The conceptual transformation from the router hierarchy to the surrogate model is shown in Fig. 2.

The functionality of the surrogate in LMS is similar to that of a replier, as used in other schemes, therefore, in the remaining sections we use the term *replier* instead of *surrogate* for consistency.

B. LMS Core Concepts

Migrating the processing from routers to repliers requires answers to the following questions:

[1]An expanded version of this work can be found in [41]. An earlier version of this work was published in [9].

[2]A best scheme would use routers for recovery.

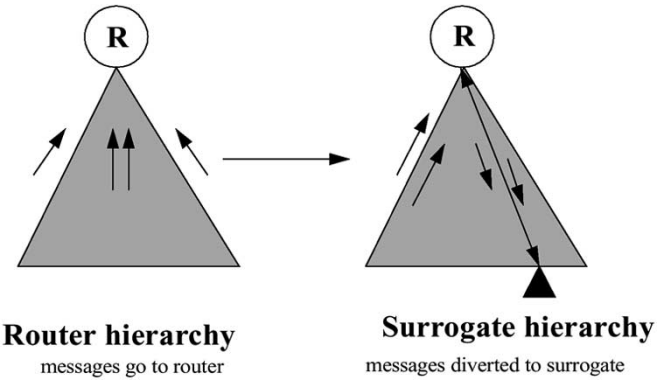


Fig. 2. LMS concept.

- How does a router select a replier?
- How does a router capture and redirect messages to its replier?
- How do repliers find their routers?
- How does the router accurately target messages from repliers to the affected receivers?

We address each of these questions in turn, but first, we provide a few clarifications in terminology. A replier is a group member selected by a router to receive requests. Many routers may select the same replier, but each router selects only one replier. A requestor is any group member that sends a request.

C. Selecting a Replier

Each router selects a single replier *for each source* in a multicast group. To simplify the description we assume that receivers are attached directly to routers (we address multiaccess networks later). Each router selects a replier as follows.

- If the router has two or more downstream links it selects one as the replier link (we will address the issue of *which* link shortly).
- If the router has only one downstream link, then that becomes the replier link by default.

As an optimization, if the source is directly attached to the router the source becomes the replier. Fig. 3 shows a possible router-replier allocation. The links leading to a replier are in bold. It is important to note that similar to data forwarding, a router only needs to know the *next hop* to the replier, not the actual replier address. For example, router R2 selects R4 as the next hop knowing that it leads to *some* replier. This has some important advantages.

- Replier changes are localized. For example, if R4 decides to switch to replier E4 (because E5 either left the group or crashed), R2 does not need to change its replier information.
- Receivers do not have to be notified when selected as repliers. A receiver knows it has been selected if it receives a request. A receiver, however, is not guaranteed to remain a replier for future requests.
- The replier state at the router is small, consisting of an identifier for the replier link.

Next, we address the replier selection criteria when a router has more than one potential replier link.

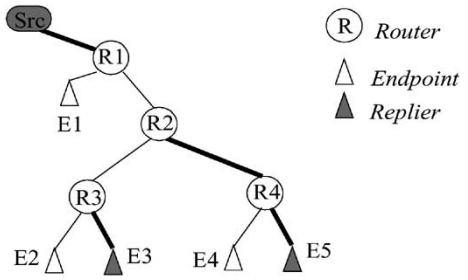


Fig. 3. Possible replier allocation in LMS.

D. Replier Selection Criteria

While it is possible to choose repliers at random, there are many reasons why we would prefer the application to drive the replier selection. For example, some receivers may be better suited to act as repliers because they have more resources or have more reliable links than others.

In LMS, receivers express their desire to become repliers by piggybacking information on the IGMP join and refresh requests. Along with each join or refresh request, receivers communicate a *cost* to the router, which is application-defined and is used to drive replier selection. Routers may select repliers by simply comparing the advertised cost. The cost semantics are transparent to LMS. 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.

E. Steering Messages to Repliers

When loss is detected, requestors multicast a request that contains a new (to be defined) IP option. Requests are handled hop-by-hop by LMS routers. Initially, routers steer requests toward the source until a replier path is found and then toward the replier. Hop-by-hop forwarding requires routers to intercept each request, which is accomplished via the IP Router Alert option [14] included in every request.

F. Request Handling at the Routers

Routers allow 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 as depicted in Fig. 4 and described below.

A request may arrive at a router from one of three possible directions.

1) **From a nonreplier link:** When a request arrives in such a manner the router becomes the *turning point* of the request. The turning point router turns requests around (recall that the request was traveling upstream until this point) and forwards them out the replier link. Before forwarding each request, and if the turning point field is empty, the router adds the following information to the packet: (a) an identifier (e.g., the IP address) of the interface the request arrived on, and (b) the IP address of the replier interface. We will see shortly how the turning point information is used. Note that the turning point globally identifies the root of the subtree where the request was generated.

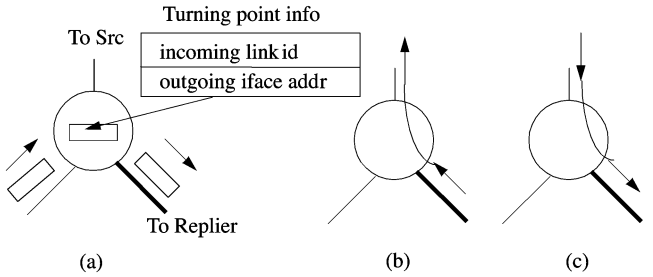


Fig. 4. Request handling at a router. (a) Request from nonreplier link. (b) Request from replier link. (c) Request from upstream link.

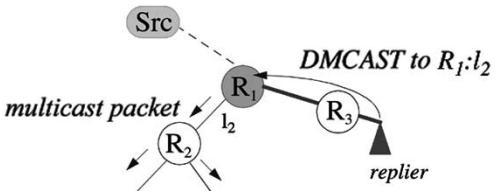


Fig. 5. Directed multicast (DMCAST).

2) **From the replier link:** When a request arrives from the replier link the router forwards it to the upstream link. The packet is not modified.
3) **From the upstream link:** When a request arrives from the upstream link the router forwards it to replier link. The packet is not modified.

It is important to note that the maximum number of requests diverted to the replier is bounded by the number of downstream links at the turning point. The replier that has the data and will service the request receives at most one request. If a replier receives a request but does not have the data, the replier ignores the request since it must have sent a similar request of its own.

G. Directed Multicast

A replier retransmits the data using a new service called Directed Multicast (DMCAST). This is the final service provided by LMS and its purpose is to enable fine-grain multicast to eliminate exposure.

The operation of a DMCAST is summarized in Fig. 5. To perform a DMCAST, a replier first creates a multicast packet containing the requested data. The source address is set to the original source and the destination address to the group. An IP option is added to the packet, containing the turning point information, which is 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 is again obtained from the request. When the turning point router receives the packet, it decapsulates the multicast packet, strips the IP option, and multicasts it on the specified interface. From there, the packet travels as if it had originated from the source.

H. LMS Summary

LMS enhances IP multicast with three important services: 1) replier selection; 2) steering requests to repliers and establishing turning points; and 3) directed multicast. These services enable receivers to construct an efficient recovery mechanism, as depicted in Fig. 6 and summarized below.

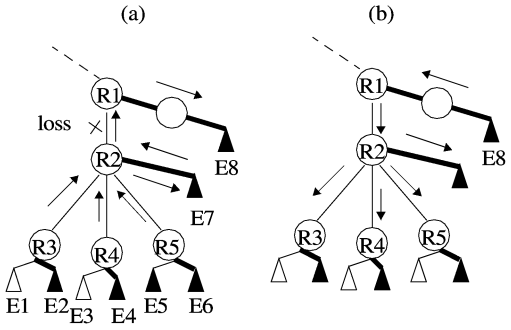


Fig. 6. LMS summary. (a) Request. (b) Reply.

1) Sending a Request: Assume a loss occurs between R1 and R2 and endpoints E1–E7 detect it. 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 Rl, which forwards it to E8, which has the requested data.

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

- E8 creates a multicast message containing the reply. E8 encapsulates the message in a unicast message and sends it to Rl (the request's turning point).
- R1 decapsulates the multicast message and multicasts it to the link leading to R2.
- All receivers downstream R2 receive the message.

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 maintain any state about passing requests. Thus, LMS requires no per-packet state at the routers. Also, since LMS packets are forwarded using the same $\langle S, G \rangle$ state as regular packets, routers need not maintain $\langle S, G \rangle$ state for many senders in single-sender groups, as in application-level recovery schemes.

I. Problem: Exposure

Since only one replier will respond to a given loss, LMS will never generate duplicate replies for the same loss. It is possible, however, that a loss on a replier link may expose other receivers to duplicates. This scenario is shown in Fig. 7. When loss occurs on a replier path, a request from replier 1 reaches replier 2, which in turn sends a directed multicast to R2. The reply is multicast on the downstream link leading to Rl, causing exposure on the branch toward R3.

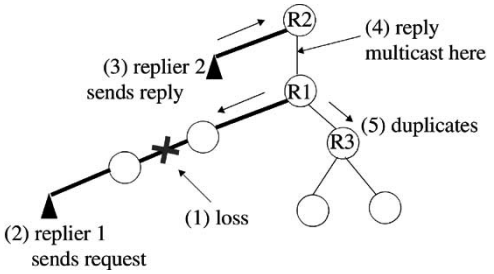


Fig. 7. Exposure in LMS.

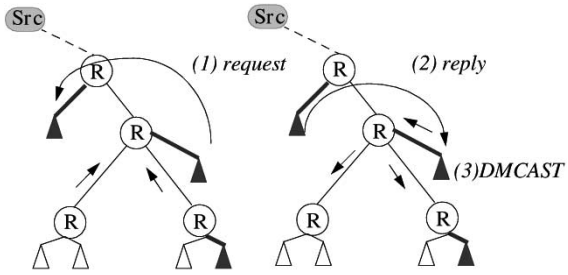


Fig. 8. Two-step recovery eliminates exposure (not used in our simulations).

One way to minimize this problem is to use the cost field to select a replier that advertises the least loss. For example, R1 will select a replier from the right-hand side if this branch experiences less loss, even though the replier on the left-hand side may be closer. Another way to address this problem is to break DMCAST into two steps as shown in Fig. 8. With this approach, the request specifies that the reply should be unicast to the requestor, rather than the turning point. If the requestor receives other requests, either while waiting for the reply or soon after, the requestor knows that this is a loss that affected more receivers and performs 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.

III. Protocol Details

In this section, we delve into some important protocol details that were not covered in the previous section.

A. Late Requests

If a request arrives at a replier after the reply was sent, it leads to the following ambiguity: is this a late request or a new request because the previous reply was lost?

To overcome this ambiguity, requestors number their requests. A replier can identify and safely ignore the first such request following a reply—obviously, the two have crossed each other. To completely eliminate the ambiguity, repliers may refrain from serving late requests until they receive a second request (triggered by a timeout). Another option is to introduce an "ignore" period at the repliers.

B. Shared Trees

LMS works well with source-based trees, such as those created by DVMRP. LMS also works with unidirectional shared trees, such as those constructed by PIM-SM, the dominant routing protocol today, as follows. Requests from repliers are directed toward the core. If a request reaches the core, it is unicast to the source. The source in turn sends a directed multicast to the core. LMS in its current state does not work with bidirectional shared trees without additional per source information. Note, however, that in applications that require source filtering, such information may be readily available.

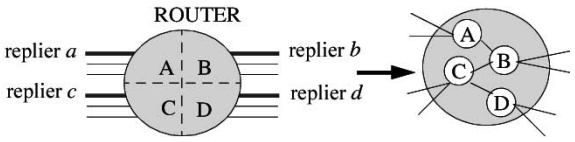


Fig. 9. Dealing with routers with large fan-out.

C. Replier Failure

Routers use soft replier state, thus, failed repliers are detected when the replier state expires. For speedier recovery, requestors may notify the turning point router after N failed requests to the current replier (where N is application-defined). Router notification requires a special flag in the request. The router (perhaps after pinging the current replier) has the option of either switching repliers immediately if a backup is available, or soliciting a new replier. During the latter, requests may be temporarily directed upstream.

D. Selecting Repliers in Multiaccess LANs

For simplicity, the previous sections assumed one receiver at each router link. In LANs, receivers use some election mechanism to elect a replier. Local receivers are responsible for monitoring the replier and triggering a new election when the replier departs or fails. The details of a possible election mechanisms as well as other issues that arise in multiaccess LANs are covered in [41].

E. Routers With a Large Fan-Out

In routers with a large fan-out, a replier may receive a large number of requests. To avoid this problem, the router may hierarchically partition its links in multiple sets as shown in Fig. 9. In this example, requests from set D go to replier d, requests from replier d go to replier c, requests from replier c go to replier b, and requests from replier a are forwarded upstream.

F. Proxy Directed Multicast

Once a request passes the turning point, it contains enough information to uniquely identify the loss subtree. Thus, if a replier cannot service a request (e.g., if the data was purged), the replier may forward it upstream but preserve the original turning-point information. This enables a directed multicast to reach the original subtree regardless of where it originates. Thus, routers must first ensure that the turning-point field is empty before overwriting it.

G. Other Applications

LMS may be used for any application that requires a hierarchy congruent with the underlying topology. One such application is a *scalable collect* service where repliers aggregate information

before forwarding upstream. The fine-grain multicast provided by the DMCAST service may be used to target specific parts of the multicast tree, for example, to announce the presence of a server in a region. LMS may implement anycast [15] as follows: servers use LMS to register as repliers; clients send LMS requests, which are directed to the nearest server. Routers advertise repliers (servers) on all links rather than just the upstream link. This technique bears some similarities to [8].

IV. Simulation Results

We evaluated the performance of LMS using simulation and compared it with two other reliable multicast schemes, namely, SRM [3] and PGM [16]. SRM is a pure application layer scheme, and thus, a performance comparison is interesting to help quantify gains, if any, of router assistance.

We implemented LMS and PGM in the UCB/LBNL/VINT network simulator ns [17]. SRM is already implemented in ns. Most of the test topologies were generated by GT-ITM [18]. Similar to the evaluation of SRM, we used artificial packet drops and measured the overhead of recovery in each scheme *after* a loss occurred. 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.

A. Evaluation Metrics

While several studies have attempted to characterize loss in the Internet [12], [19], their results have not been conclusive. Thus, to avoid making our own (most likely flawed) assumptions about loss in multicast networks, we limit our study to just two performance metrics: latency and exposure. A similar set of metrics was used to evaluate SRM.

We defined the following metrics: *normalized recovery latency* (for all schemes); *exposure* (for LMS); *requests/repairs per drop* (for SRM); and *repeated retransmissions per drop* (for PGM). The definitions of these metrics are shown in Fig. 10.

Normalized recovery latency is defined as the latency that a receiver experiences from the moment it detects a loss until the loss is recovered, divided by the receiver's round-trip time (RTT) 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 apply to PGM and may occur due to retransmissions triggered by nearby receivers erasing router state before all links are grafted into the retransmission tree (see Section IV-H).

We used three types of topologies in our simulations: binary trees, random topologies, and transit–stub topologies. Random and transit–stub topologies were generated with the Georgia Tech Internet Topology Models (GT-ITM). While binary trees

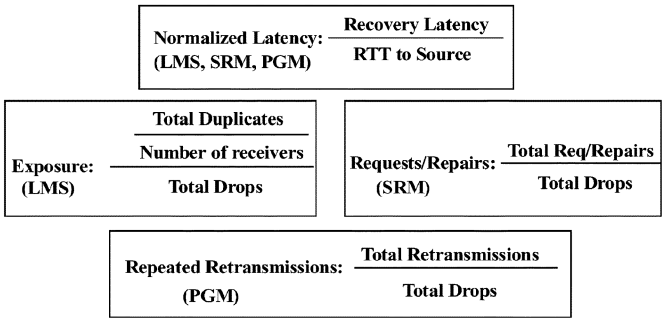


Fig. 10. Evaluation metrics.

are a topology unlikely to be encountered frequently or at a large scale in the Internet, they represent 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.

B. Simulation Parameters

For binary trees, we simulated trees of height ranging from three to seven (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 ten random and ten transit–stub topologies, each containing 100 nodes. We ran simulations with 5, 20, and 100 receivers, randomly distributed over the internal nodes. For each topology, we ran ten simulations, each with a different receiver allocation (generated by randomly seeding 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.

Similar to the study of SRM, our simulation runs used a single packet loss. Loss location has significant impact on performance, thus, we chose to investigate the following three cases.

- Loss at the source: a packet is lost near the source such that all receivers miss it. This case tests the scheme's ability to control NACK implosion.
- Loss at a 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 a 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.

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

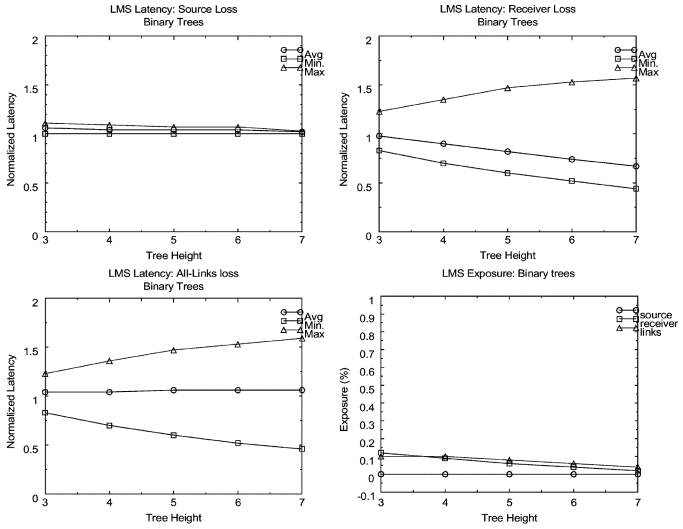


Fig. 11. LMS with binary trees.

C. LMS Experiments: Static Versus Dynamic Repliers

The repliers in LMS experiments were static, selected at the beginning of the simulation to minimize latency. Despite the potentially severe performance limitations that static repliers may entail, we opted for static repliers for two reasons: 1) without knowledge of the loss characteristics of the network, it is hard to devise an efficient replier adaptation scheme; and 2) 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 plays the most important role.

D. Binary Trees

The first experiment uses binary trees with loss at the source. The results are shown in Fig. 11. We plot the average, minimum, and maximum recovery latency. We do not plot error bars 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 one. The minor deviations are caused by a 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 toward 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 source, thus, stretching the recovery latency beyond one. Note, however, that in binary trees the normalized recovery latency can never exceed two. The reason is that the maximum distance (in number of hops) between any two receivers is given by the expression $d = 20 \cdot (h - 1)$. As the tree height increases, the maximum normalized latency becomes

$$\lim_{h \to \infty} \frac{h-1}{h} = 2.$$

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 one. 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 appears to perform quite well. It recovers losses in about one RTT, and keeps exposure very low.

E. Random Topologies

The random topologies in our experiments were generated with GT-ITM. Edges were created uniformly with probability 0.1. Topologies consist of 100 routers with 100 randomly assigned receivers and a source, for a total of 201 nodes.

The first experiment measures recovery latency. The top-left graph in Fig. 12 shows results with loss at the source. The term R100-100 denotes "random graph with 100 routers 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 routers. The maximum latency is again around one, experienced by receivers that are close to the source.

Next, 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 distances greater than the source, a situation also seen with binary trees. With loss at all links, the results do not change significantly. The average latency increases slightly to about 60% and minimum and maximum latencies are virtually unchanged.

The next set of experiments show exposure levels to be 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 lower with random graphs than with binary trees, confirming our previous claim that binary trees are a difficult topology for LMS.

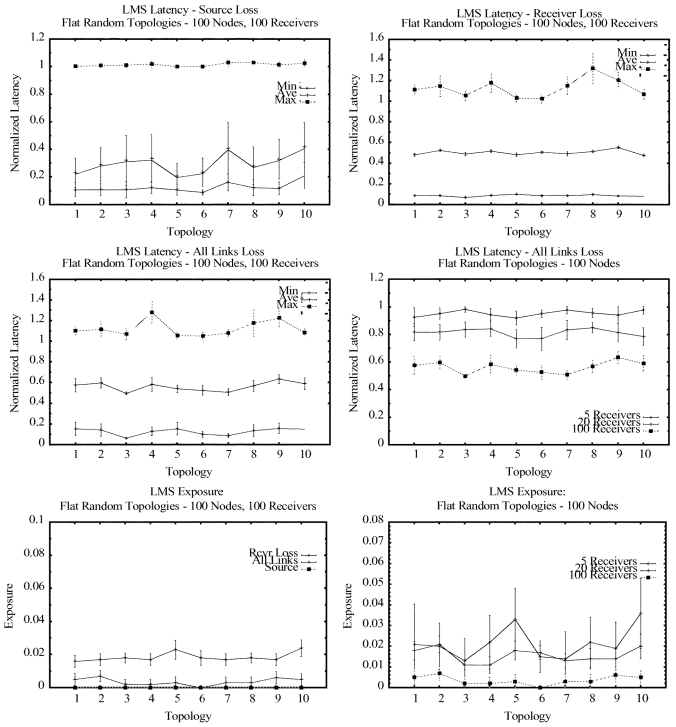


Fig. 12. LMS with random topologies.

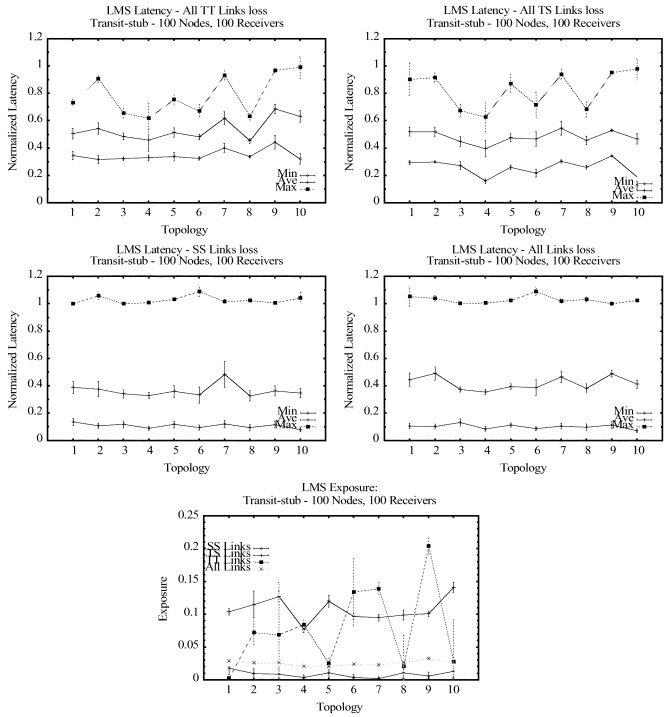


Fig. 13. LMS transit–stub topologies.

Sensitivity: Sparse Groups: In this experiment, we test the performance of LMS with sparse groups. We used the same topologies as in the previous experiments, altering only the number of receivers. The results are shown in Fig. 12. We plot simulation results with 5, 20, and 100 receivers. Loss is at all links and we measure average latency only. From the results, we notice that the recovery latency is inversely proportional to the number of receivers, which 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 of finding a helper better located to serve retransmissions without causing exposure. Performance improvements with larger groups are also seen with SRM and PGM.

F. Transit–Stub Topologies

In this section, we examine the performance of LMS with transit–stub topologies, which are a better approximation of the hierarchical structure of the Internet.

As with random topologies, we generated ten 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 four transit nodes; each transit node had three transit–stub nodes; and each transit–stub node had eight 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 assigned 100 receiver uniformly, but in this case the receivers were assigned to stub nodes only. Unlike random topologies, rather than simulating loss at the source and receivers, we simulate loss at the different types of links. Thus, in addition to loss at all links, we studied loss at transit–transit, transit–stub, and stub–stub links.

The results are shown in Fig. 13. While latency remains at about 50% on the average when all links are lossy, latency increases slightly for loss at higher levels (transit–transit links) and less so at middle levels (transit–stub links). 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.

G. SRM Experiments

SRM employs two clever global mechanisms to limit the number of recovery messages, 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 sent. The intended goal is to allow the multicast of only one or a few recovery messages. In order to increase the effectiveness of the suppression mechanism, especially in densely packed groups, the RTT 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 from 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

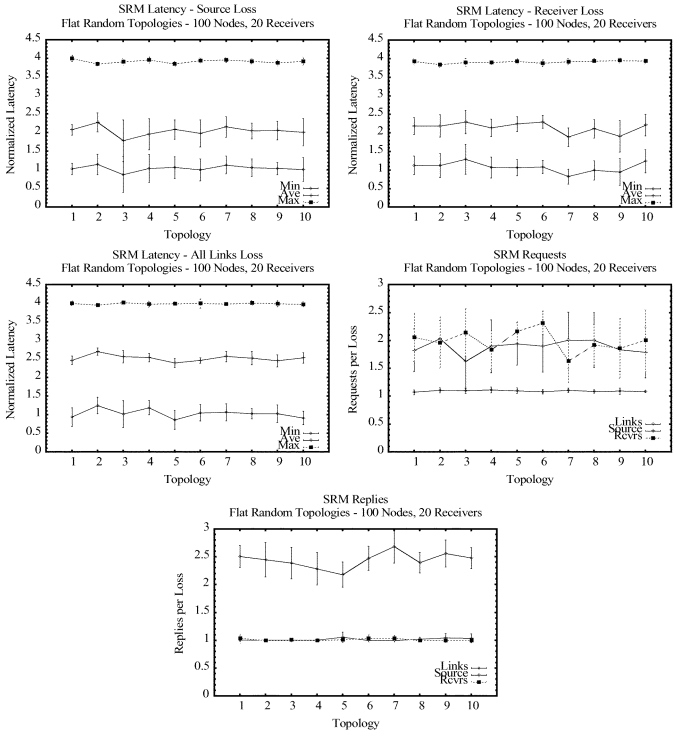


Fig. 14. SRM with random topologies.

recalculated when group membership, topology, or network conditions change, meaning that SRM needs time to adapt.

SRM has already been extensively studied via simulation and results have been reported elsewhere [3]. Our goal here is 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. The reason SRM simulations are slow is that the SRM implementation in ns is done mostly in Tcl.

Our results are consistent with results presented previously. In the following sections, we present results from simulating SRM in random topologies only. The random topologies are the same used with LMS, but with 20 receivers. We report results for normalized latency and the number of requests and replies generated for each lost packet.

Fig. 14 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 two RTTs, or twice the unicast latency, with the maximum value being around four and the minimum around one. We also note that the recovery latency is relatively uniform over all topologies. We believe the reason is that the back-off timers absorb any differences that may arise due to a particular topology. In addition, SRM appears to be insensitive to loss location.

Looking at generated requests, the linear component in the back-off timers works well and keeps the number of requests low. The results, however, are significantly worse for replies where SRM may generate four to five replies for each loss. For

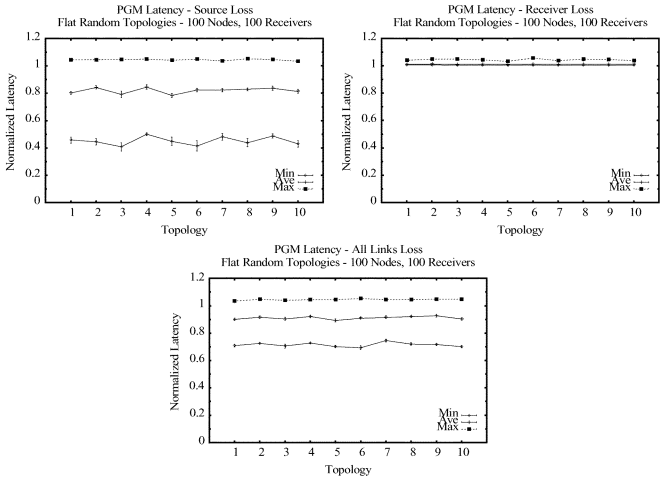


Fig. 15. PGM with random topologies.

replies, the linear component is less effective since the inter-receiver RTT is smaller in general than each receiver's RTT to the source.

H. PGM Experiments

PGM [16] is a reliable multicast protocol marketed by the router company Cisco. PGM is a network-assisted scheme, that requires per-lost-packet state at the routers. In PGM, NACKs create state at the routers to avoid sending duplicate NACKs 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).

In PGM, NACKs are propagated reliably hop-by-hop and only one NACK reaches the sender. In addition, state left behind at the routers ensures that a receiver will not get a retransmission unless it has sent a NACK. There are cases, however, where a single retransmission by the sender will reach all receivers that have requested it and PGM may have to send the same retransmission multiple times. This happens when a nearby receiver sends a NACK and triggers a retransmission before NACKs from distant receivers establish NACK state in downstream routers. Since NACK state is wiped out by retransmissions, a NACK arriving at a router after a retransmission has passed will re-establish NACK state back to the source. This is called the *repeated retransmissions problem*. We examine the impact of this problem in our simulations. The PGM specification proposes that the sender waits for an interval up to the maximum RTT of any member in the group before sending a retransmission to avoid this problem.

The topologies and parameters used for the PGM experiments are the same as with LMS and SRM. Our PGM simulation did not include all the features described in the PGM specification. However, we believe our simulation includes enough functionality to capture the basic operation of PGM.

Fig. 15 shows the PGM results. With loss at the source, average recovery takes about 80% of the unicast latency. This experiment did not produce results similar to LMS due to repeated retransmissions, which delayed recovery a little.

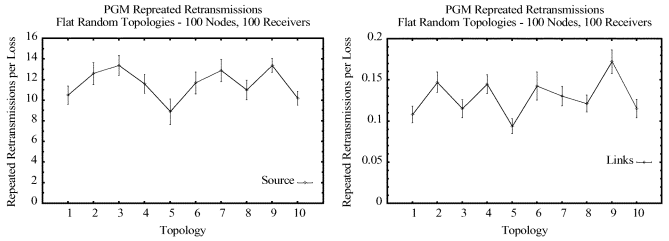


Fig. 16. PGM repeated retransmissions.

TABLE I
PERFORMANCE SUMMARY

Scheme	Latency	Exposure	Repeated RTX
LMS	30 - 60%	0.5%	none
SRM	>200%	4-6 per loss	none
PGM	80 - 100%	none	<10-13 per loss

With loss near the receivers the recovery latency is very close to one, as expected. Note that with PGM, recovery latency does not increase much beyond one because a retransmission always comes from the source. With loss at all links, recovery latency in PGM increases slightly to about 90% of unicast latency.

We ran experiments to estimate the number of repeated retransmissions when loss occurs near the source, which is the worst case scenario. Our results are shown in Fig. 16, and they show that the number of repeated retransmissions can reach 9–13 per lost packet when there is no wait interval at the source, which suggests that the use of such interval will be required.

I. Discussion

The performance results of all three protocols are summarized in Table I. In general, it appears that network support offers an advantage for LMS and PGM, both of which show improvements over SRM. Improvements are apparent in both recovery latency and exposure. An additional benefit is that network assistance frees the protocol from maintaining topology-related information, which is hard to estimate.

Comparing LMS and PGM, we note that LMS is much simpler to implement at the routers, yet its performance is on par with PGM. LMS has lower recovery latency because it enlists help from all receivers, at the expense of slightly higher exposure. PGM requires per-lost-packet state at the routers which may be significant for large backbone routers.

V. LMS PROCESSING OVERHEAD

We implemented LMS in the IP networking stack of Net-BSD and evaluated the processing overhead using the test-bed shown in Fig. 17. We measured the forwarding overhead at the LMS router, but did not measure the additional processing at the hosts because our changes there were minimal. Measurements were done on 300-MHz Pentium II machines and a 155-Mb/s ATM network. The multicast sender is on the host marked SRC and hosts H1 and H2 are receivers. Host H1 (shaded) is the replier. The measurements were taken using the processor cycle counter register in the Pentium processor. We measured the processing

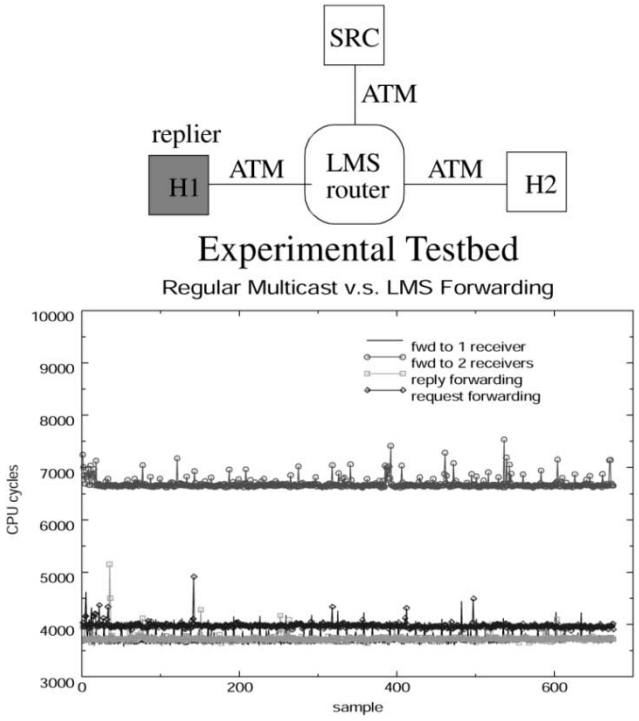


Fig. 17. Experimental testbed and LMS forwarding cost.

TABLE II
NORMAL VERSUS LMS FORWARDING COST

IP mcast 1 receiver	IP mcast 2 receivers	LMS request	LMS dmcast
3702 cycles	6686 cycles	3979 cycles	3734 cycles
12.3 μs	22.3 μs	13.3 μs	12.4 μs

at the entire IP layer, from the moment a packet was received at IP until the packet was passed to the network interface.

First, 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 the first two columns of Table II.

Next, we measured the processing overhead of LMS with two experiments, one measuring the processing overhead to forward a request, and the other 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 are shown in the right two columns of Table II.

The results show 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 view of the results is shown in Fig. 17.

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 multicast packet. It appears that the forwarding cost of regular multicast packets increases almost linearly as the router member interfaces increase. 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, the cost remains constant regardless of the router fan-out. This shows that LMS processing at the routers is *not* a bottleneck.

VI. Related Work

Given the richness of multicast applications, a general transport service has proven difficult to design. Recognizing these issues, the research community has focused toward developing and standardizing a set of multicast building blocks [40] intended to be combined into customized services. In this section we first summarize recent activity on alternate multicast service models and then summarize the large body of work on reliable multicast, including end-to-end and network assisted schemes.

Alternate Multicast Service Models: EXPRESS [36] proposes a service model where only one source is allowed to transmit and requires receivers to explicitly join. This solves the rendezvous problem but trades anonymity for access control and billing. EXPRESS is geared toward highly populated one-to-many channels like Internet TV. Simple Multicast [37] solves the rendezvous problem by including the address of the core in the address of the group. Unlike EXPRESS, SM creates bidirectional shared trees. Both approaches solve the address scarcity problem by allowing 24 bits of multicast addressing per router. Changes to the service model such as Single-Source Multicast (SSM) [47] have also been proposed, which better align with certain user needs.

End-to-End Reliable Multicast: Early work on reliable multicast has focused on distributed systems and includes ISIS [23] and the V-kernel [24]. Other early work focused on multi-access local area networks [25]–[28]. A survey of the early work can be found in Levine [20].

Pingali *et al.* [10] showed that receiver reliable multicast is superior to sender-reliable recovery. RMTP [4] constructs a static hierarchy, where the source multicasts data to all receivers, but only *designated receivers* (DRs) return acknowledgments. Although not implemented, RMTP was the first protocol to propose the use of *subcast*, a service similar to directed multicast.

The Log-Based Receiver-reliable Multicast (LBRM) [7] uses a primary logging server and a static hierarchy of secondary logging servers which retransmit lost data. The Tree-based Multicast Transport Protocol (TMTP) [13] uses a dynamic hierarchy of domain managers (DMs) using an expanding ring search. Each endpoint maintains the hop distance to its DM, and each DM maintains the hop distance to its farthest child. TMTP also uses randomized backoff for requests. In LGMP [29] receivers dynamically organize themselves into subgroups and select a Group Controller. TRAM [31] uses TTL to form the receiver tree and the tree formation and maintenance algorithms borrow from schemes such as TMTP, but with richer tree management. MFTP [30] transmits data in rounds. After each round, receivers unicast NACKs back to the sender, which collects all NACKs and transmits all missing packets in the next round. Kasera presents two schemes for reliable multicast, one using multiple multicast channels [45] and another using active services [46].

Forward error correction (FEC) is an alternative error control scheme, which works well in environments with a high degree of uncorrelated loss. FEC typically increases the bandwidth required to transmit data. Recent techniques reduce this overhead and increase the effectiveness of FEC in multicast [32]–[34].

Network Assisted Schemes: Addressable Internet Multicast (AIM) [8] assigns per-multicast group labels to all routers participating in a group. There are three types of labels: positional, distance, and stream labels. Positional labels route messages to individual members of the group. Distance labels help locate nearby members and stream labels are used to subscribe to specific sources. Search Party [6] routes requests to the parent or one of the children using "randomcast." Search Party trades some efficiency for robustness.

OTERS [35] uses a modified version of mtrace [22] to construct recovery trees congruent with the underlying multicast tree. Tracer [21] is similar to OTERS in that it also uses mtrace to allow each receiver to discover its path to the source. Then, receivers share path information with near-by receivers using expanding ring search and select parents based on path and loss information. Active Reliable Multicast [44] uses active routers for recovery.

More recent work includes a comparison study of the costs of application-layer reliable multicast schemes and router-assisted schemes [42]. Finally, He *et al.* [43] present a comparison study between incremental deployment of LMS and PGM.

VII. Conclusion

In this paper, we have presented LMS, a set of forwarding services that enhance the IP multicast service model to allow the implementation of highly scalable reliable multicast applications. We have shown that LMS is simple to implement and its overhead at the routers is comparable to normal multicast. We have demonstrated through simulation that LMS significantly improves the performance of multicast error recovery compared to pure end-to-end schemes such as SRM. In addition, LMS greatly simplifies the implementation of multicast applications by eliminating the burden of maintaining topology-related state. LMS can also be used in other allocations, such as anycast and a scalable collect service.

A novel contribution of this work is the decomposition of transport and forwarding functionality of multicast error control, such that each can be located where it is most beneficial. This separation is very clean in that it does not violate any layering principles. This may have far-reaching implications in that it provides us with a new vantage point from where other important yet difficult multicast problems can be viewed, such as congestion control, ACK-based reliable multicast, and topology-aware grouping.

ACKNOWLEDGMENT

The authors would like to thank S. Shi for developing the ns code for PGM and for long fruitful discussions.

REFERENCES

[1] A. Tanenbaum, *Computer Networks*. Englewood Cliffs, NJ: Prentice-Hall, 1988.

[2] D. Comer, *Internetworking with TCP/IP*. Englewood Cliffs, NJ: Prentice-Hall, 1991.

[3] S. Floyd, V. Jacobson, S. McCanne, C. Liu, and L. Zhang, "A reliable multicast framework for light-weight sessions and application framing," in *Proc. ACM SIGCOMM*, Cambridge, MA, 1995, pp. 342–356.

[4] J. Lin and S. Paul, "RMTP: A reliable multicast transport protocol," *Proc. IEEE INFOCOM*, pp. 1414–1424, Mar. 1996.

[5] S. Deering, "Host Extensions for IP Multicasting," Network Working Group, RFC 1112, Jan. 1989.

[6] A. Costello and S. McCanne, "Search party: Using randomcast for reliable multicast with local recovery," in *Proc. IEEE INFOCOM*, New York, NY, Mar. 1999, pp. 1256–1264.

[7] H. Holbrook, S. Singhal, and D. Cheriton, "Log-based receiver-reliable multicast for distributed interactive simulation," in *Proc. ACM SIGCOMM*, vol. 25, Oct. 1995, pp. 328–341.

[8] B. Levine and J. J. Garcia-Luna-Aceves, "Improving Internet multicast with routing labels," in *Proc. IEEE Int. Conf. Network Protocols*, Atlanta, GA, Oct. 1997, pp. 241–250.

[9] C. Papadopoulos, G. Parulkar, and G. Varghese, "An error control scheme for large-scale multicast applications," in *Proc. IEEE INFOCOM*, San Francisco, CA, Mar. 1998, pp. 1188–1196.

[10] S. Pingali, D. Towsley, and J. Kurose, "A comparison of sender-initiated and receiver-initiated reliable multicast protocols," in *J. Select. Areas Commun.*, vol. 15, Apr. 1997, pp. 398–406.

[11] J. H. Saltzer, D. P. Reed, and D. D. Clark, "End-to-end arguments in system design," *ACM Trans. Comput. Syst.*, vol. 2, no. 4, pp. 277–288, Nov. 1984.

[12] M. Yajnik, J. Kurose, and D. Towsley, "Packet loss correlation in the MBONE multicast network: Experimental measurements and Markov chain models," in *Proc. IEEE GLOBECOM*, Nov. 1996, pp. 94–99.

[13] R. Yavatkar, J. Griffioen, and M. Sudan, "A reliable dissemination protocol for interactive collaborative applications," in *Proc. ACM Multimedia*, 1995, pp. 333–344.

[14] D. Katz, "IP Router Alert Option," Network Working Group, RFC 2113, Feb. 1997.

[15] C. Partridge, T. Mendez, and W. Milliken, "Host Anycasting Service," Network Working Group, RFC 1546, Nov. 1993.

[16] T. Speakman, J. Crowcroft, J. Gemmell, D. Farinacci, S. Lin, D. Leshchiner, M. Luby, T. Montgomery, L. Rizzo, A. Tweedly, N. Bhaskar, R. Edmonstone, R. Sumanasekera, and L. Vicisano, "PGM Reliable Transport Protocol Specification," IETF, RFC 3208, Dec. 2001.

[17] UCB/LBNL/VINT Network Simulator-ns (Version 2). [Online]. Available: http://www-mash.cs.berkeley.edu/ns/

[18] E. Zegura, K. Calvert, and S. Bhattacharjee, "How to model an Internetwork," in *Proc. IEEE INFOCOM*, San Francisco, CA, Mar. 1996, pp. 594–602.

[19] M. Handley, "An Examination of MBone Performance," Information Sciences Institute, University of Southern California, Los Angeles, USC/ISI Research Report ISI/RR-97-450, Jan. 1997.

[20] B. N. Levine, D. Lavo, and J. J. Garcia-Luna-Aceves, "The case for concurrent reliable multicasting using shared ack trees," in *Proc. ACM Multimedia*, Boston, MA, Nov. 1996, pp. 365–376.

[21] B. N. Levine, S. Paul, and J. J. Garcia-Luna-Aceves, "Organizing multicast receivers deterministically according to packet-loss correlation," in *Proc. ACM Multimedia*, Bristol, U.K., Sept. 1998, pp. 201–210.

[22] W. Fenner and S. Casner, "A "traceroute" facility for IP Multicast," IETF, Internet Draft, draft-fenner-traceroute-ipm-00.txt, Dec. 2003.

[23] K. Birman and T. Joseph, "Reliable communication in the presence of failures," *ACM Trans. Comput. Syst.*, vol. 5, no. 1, pp. 47–76, Feb. 1987.

[24] D. Cheriton and W. Zwaenepoel, "Distributed process groups in the V kernel," *ACM Trans. Comput. Syst.*, vol. 3, no. 2, pp. 77–107, May 1985.

[25] J. Chang and N. Maxemchuck, "Reliable broadcast protocols," *ACM Trans. Comput. Syst.*, vol. 2, no. 3, pp. 251–273, Aug. 1984.

[26] M. Kaashoek, A. Tanenbaum, S. Humel, and H. Bal, "An efficient reliable broadcast protocol," *ACM Oper. Syst. Rev.*, vol. 23, no. 4, pp. 5–19, Oct. 1989.

[27] L. C. N. Tseung, "Guaranteed, reliable, secure, broadcast networks," *IEEE Network*, vol. 3, pp. 33–37, Nov. 1989.

[28] J. Crowcroft and K. Paliwoda, "A multicast transport protocol," in *Proc. ACM SIGCOMM*, Aug. 1988, pp. 247–256.

[29] M. Hofmann, "A generic concept for large scale multicast," in *Proc. Int. Zurich Seminar on Digital Communications*, Zurich, Switzerland, Feb. 1996, pp. 95–106.

[30] K. Miller, K. Robertson, A. Tweedly, and M. White, "StarBurst Multicast Transfer Protocol (MFTP) Specification," IETF, Internet Draft, draft-miller-mftp-spec-03.txt, Jan. 1997.

[31] D. Chiu, S. Hurst, M. Kadansky, and J. Wesley, "TRAM: A Tree-Based Reliable Multicast Protocol," Sun Microsystems, Santa Clara, CA, Sun Tech. Rep. SML TR-98-66, July 1998.

[32] L. Vicisano and J. Crowcroft, "One to many reliable bulk data transfer on the MBone," presented at the 3rd Int. Workshop High Performance Protocol Architectures (HIPPARCH'97), Uppsala, Sweden, June 1997.

[33] J. Nonnenmacher, E. Biersak, and D. Towsley, "Parity-based loss recovery for reliable multicast transmission," *IEEE/ACM Trans. Networking*, vol. 6, pp. 349–361, Aug. 1998.

[34] D. Rubenstein, J. Kurose, and D. Towsley, "Real-time reliable multicast using proactive forward error correction," presented at the NOSSDAV'98, Cambridge, U.K., July 1998.

[35] D. Li and D. R. Cheriton, "OTERS (On-tree efficient recovery using subcasting): A reliable multicast protocol," in *Proc. 6th IEEE International Conf. Network Protocols (ICNP'98)*, Austin, Texas, Oct. 1998, pp. 237–245.

[36] H. Holbrook and D. Cheriton, "IP multicast channels: EXPRESS support for large-scale single-source applications," in *Proc. ACM SIGCOMM*, Cambridge, MA, Aug. 1999, pp. 65–78.

[37] R. Perlman, C.-Y. Lee, A. Ballardie, J. Crowcroft, Z. Wang, T. Maufer, C. Diot, J. Thoo, and M. Green, "A Design for Simple, Low-Overhead Multicast," IETF, Internet Draft, draft-perlman-simple-multicast-03.txt, Oct. 1999.

[38] R. Comerford, "The state of the Internet: Roundtable 4.0," *IEEE Spectrum*, vol. 35, pp. 69–79, Oct. 1998.

[39] C. Diot, B. Levine, B. Lyles, H. Kassem, and D. Balensiefen, "Deployment issues for the IP multicast service and architecture," *IEEE Network*, vol. 14, pp. 78–88, Jan./Feb. 2000.

[40] B. Whetten, L. Vicisano, R. Kermode, M. Handley, S. Floyd, and M. Luby, "Reliable Multicast transport building blocks for one-to-many bulk-data transfer," IETF, RFC 3048, Jan. 2001.

[41] C. Papadopoulos, "Error control for continuous media and large scale multicast applications," Ph.D. thesis, Dept. Comput. Sci., Washington Univ., St. Louis, MO, Aug. 1999.

[42] P. Radoslavov, C. Papadopoulos, R. Govindan, and D. Estrin, "A comparison of application-level and router-assisted hierarchical schemes for reliable multicast," in *Proc. IEEE INFOCOM*, Anchorage, AK, 2001, pp. 229–238.

[43] X. He, C. Papadopoulos, and P. Radoslavov, "A framework for incremental deployment strategies for router-assisted services," in *Proc. IEEE INFOCOM*, San Francisco, CA, 2003, pp. 1488–1498.

[44] L. Lehman, S. Garland, and D. Tennenhouse, "Active reliable multicast," in *Proc. IEEE INFOCOM*, San Francisco, CA, 1998, pp. 581–589.

[45] S. Kasera, G. Hjalmtysson, D. Towsley, and J. Kurose, "Scalable reliable multicast using multiple multicast channels," *IEEE/ACM Trans. Networking*, vol. 8, pp. 294–310, June 2000.

[46] S. K. Kasera, S. Bhattacharyya, M. Keaton, D. Kiwior, J. Kurose, D. Towsley, and S. Zabele, "Scalable fair reliable multicast using active services," *IEEE Network*, vol. 14, pp. 48–57, Jan./Feb. 2000.

[47] H. Holbrook and B. Cain, "Source-Specific Multicast for IP," IETF, Internet draft, draft-ietf-ssm-arch-04.txt, Oct. 2003.

Christos Papadopoulos (M'92) received the Ph.D. degree from Washington University, St. Louis, MO, in 1999.

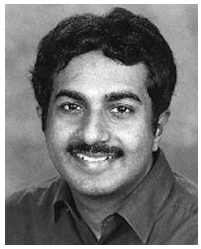
He is currently an Assistant Professor at the University of Southern California (USC), Los Angeles, where he does research on network security and multimedia communications. He is also affiliated with the Information Sciences Institute (ISI) and the Integrated Media Systems Center (IMSC) at USC.

Dr. Papadopoulos received the National Science Foundation Career Award in 2002. He has been a member of the Association for Computing Machinery since 1999.

Guru Parulkar received the Ph.D. degree in computer science from the University of Delaware, Newark, in 1987.

He has been in the field of networking for nearly 20 years and has worked in academia, startups, a large company, and a venture capital firm. He recently joined the National Science Foundation as a Program Director in its new Computer and Networking Systems Division, and the University of California, Riverside, as a Professor of computer science. He was a cofounder, CTO, and Member of the Board of Growth Networks (acquired by Cisco Systems). Prior to that, he was a Professor of computer science and Director of the Applied Research Laboratory at Washington University, St. Louis, MO.

Dr. Parulkar is a recipient of an Alumni Outstanding Achievement Award and a Frank A. Pehrson Graduate Student Achievement Award in Computer and Information Sciences from the University of Delaware, and an Entrepreneurship Award from the National Endowment for the Arts (NEA).



George Varghese (M'89) received the Ph.D. degree from the Massachusetts Institute of Technology, Cambridge, in 1992.

He is a Professor in the Computer Science Department, University of California at San Diego, where he does research in designing reliable protocols and efficient protocol implementations.

Dr. Varghese is a Fellow of the Association for Computing Machinery.