Scalability of Two Reliable Multicast Protocols

Oznur Ozkasap, Zhen Xiao and Kenneth P. Birman[1]

Dept. of Computer Science

Cornell University, Ithaca, New York

***Abstract*—Growing demand for multicast communication in large network settings has focused attention on the scalability of reliable multicast protocols. Our paper uses both simulation tools and experiments to compare two scalable protocols, focusing on an aspect not often studied: we emphasize stability of latency distributions as these protocols scale, although also considering overhead and link utilization. These properties are considered in a variety of network topologies and with several levels of packet loss. Our findings confirm that SRM scales poorly under some conditions: to obtain reliability, the protocol incurs overhead linear in group size and throughput fluctuates erratically. We also show that SRM latencies can be very large and that latency distributions are unstable as a function of group size and network topology. Our own protocol, Bimodal Multicast, also exhibits overhead growth, but the rate of growth is slow, and latency distributions and delivery throughput rates are stable.**

***Index terms*—scalable reliable multicast, bimodal multicast, throughput stability, SRM, pbcast.**

A. INTRODUCTION

This paper explores the scalability of two reliable multicast protocols under a variety of realistic conditions, using both simulation and experiments. Our interest is in the growth of overhead and the distribution of message delays as a function of network size, when low levels of packet loss or short bursts of packet loss occur. The levels of perturbation we consider are believed to be typical of real-world networks under normal conditions. We find that even low levels of packet loss can provoke non-scalable effects, such as growth in background overhead proportional to the size of the multicast group, or extremely erratic and bursty throughput rate fluctuations. In networks divided into two clusters connected by a lossy distance link, we find very long packet delays in one of the protocols.

The first protocol considered is Scalable Reliable Multicast (SRM), a well-known reliable multicast protocol originally developed as part of the Wb application for the mbone, and subsequently extended into a free-standing reliable data transport for the internet [6]. Our work with SRM made use of an NS-2 [1] simulation, developed by the SRM designers. Both the standard and the adaptive SRM versions were studied. The second protocol we investigate is Bimodal Multicast[2], also known as "pbcast". We investigate pbcast under the same conditions as SRM, but also use an implementation to validate our findings in an experimental setting. For brevity, our paper limits itself to a very terse description of each of the protocols. The bulk of our material is concerned with the data we obtained in our studies. The paper concludes with a review of prior work, including some prior studies of SRM scalability [7,9,10,11], and offers some general observations about the challenges confronting reliable multicast protocols in large networks.

B. PROTOCOL DESCRIPTIONS

SRM Protocol (Scalable Reliable Multicast)

SRM is a reliable multicast protocol which makes receivers responsible for recovery when network packet loss occurs. The protocol makes extensive use of IP multicast. The sender and receivers join an IP multicast group, and new messages are transmitted using IP multicast, an unreliable protocol. A receiver that detects data loss uses IP multicast to solicit a retransmission, and a participant receiving a solititation uses IP multicast to repair the loss.

In SRM, the application has responsibility for "framing" the data: determining what data to send in each message, and what data to use in repairing losses. The reasoning is that the best way to overcome data loss is application-dependent. Accordingly, SRM is fairly tightly bound to the application, using upcalls and downcalls with which it reports events to the application, and is informed of the application's desired response at each participating machine.

The major innovation of SRM involves its use of stochastic mechanisms to avoid storms of solicitations and repairs when loss occurs. For example, a randomized delay is introduced before sending a solicitation or repair, and the size of the delay is increased as a function of the estimated distance of the receiver from the sender. If a process p is waiting to solicit a retransmission for lost data, it will inhibit its own request in the event that a solicitation from process q is received first. Similarly, a repair sent by one process will inhibit the sending of a repair by some other process. The time-to-live field of the IP multicast protocol is employed to limit the scope of solicitations and repairs, in the hope of repairing problems where they occur. To ensure that lost data will be detected, all members of an SRM group send "session" messages periodically, at a frequency calibrated to keep the background overhead low (less than 5%).

Our simulations consider two versions of SRM: the basic protocol and an adaptive version [6], which adjusts protocol parameters dynamically based on observations of network dynamics. Both are available from the for use in NS-2. Optimizations supporting "local recovery" and scalable session messages were not considered. Although it is plausible that we could have added these protocols extensions, or evaluated other protocols, we felt concern that the resulting simulations might not be fair to the original proposals. In our

[1] This work was supported by DARPA/ONR contracts N0014-96-1-10014 and the Turkish Research Foundation.
Email addresses: {ozkasap,xiao,ken}@cs.cornell.edu

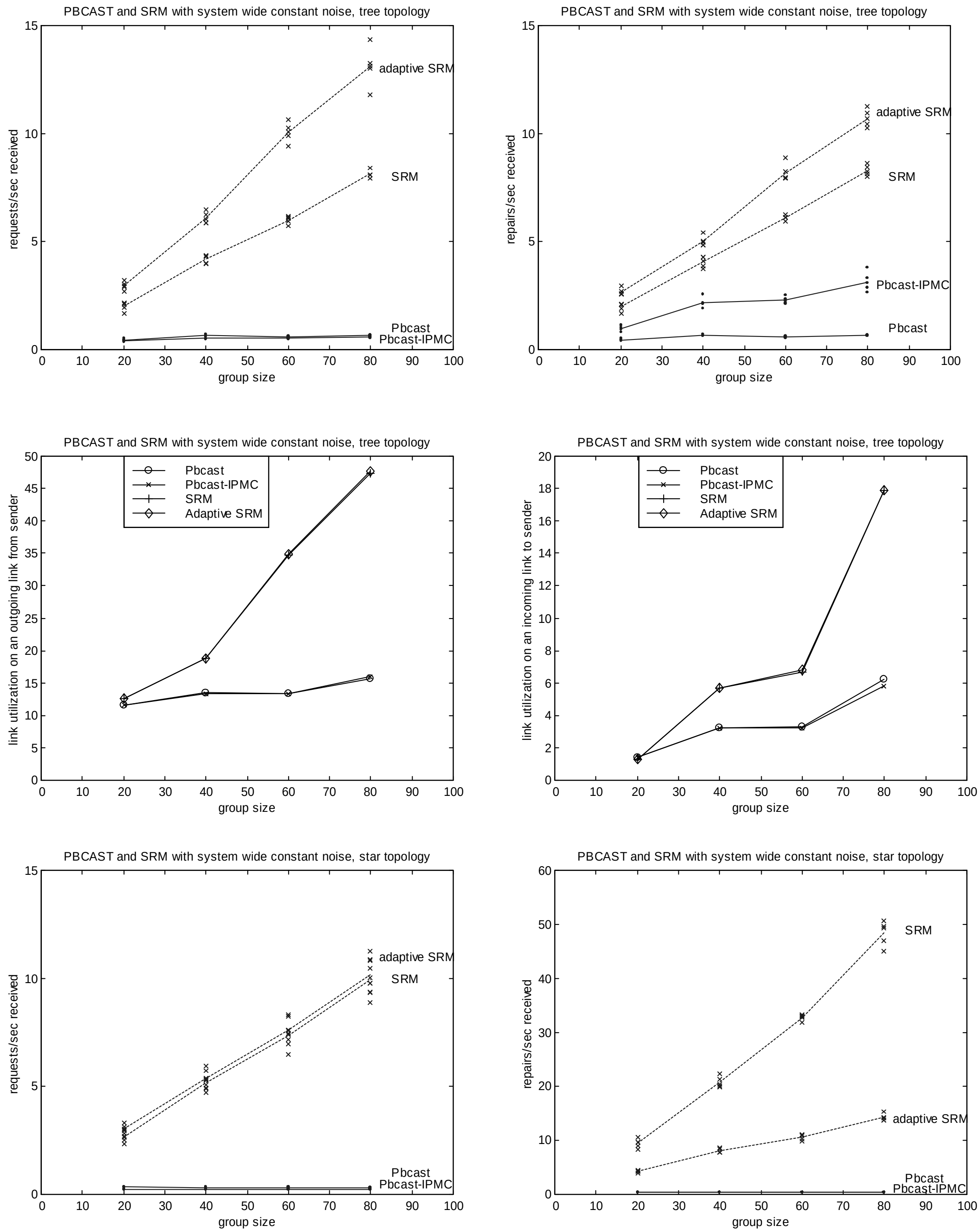

Figure 1: Overhead of pbcast and SRM. Top: retransmission requests and repairs per second, tree topology with 0.1% noise. Middle: link utilization. Bottom: requests and repairs in a star topology.

experience, only the developer of a protocol is really in a position to develop a high quality implementation.

.

Pbcast Protocol (Bimodal or "Probabilistic" Multicast)

Pbcast emerges from a flurry of research on gossip protocols [2,4,5,8,12,13]. In contrast to SRM, which provides best-effort reliability, pbcast provides a form of reliability that can be rigorously quantified [2]. This takes the form of a bimodal probability curve: with very high probability, it will reach almost all destinations, and with very small probability, a pbcast message may reach very few of its destinations. Protocol parameters can be adjusted so that other outcomes will be of negligible probability. Furthermore, pbcast has a very high probability of providing steady throughput even if packets are lost or some group members fail or behave erratically. By stable throughput, we mean predictable and small varience in the data delivery rate, under conditions where data is generated at a steady rate. Unlike SRM, pbcast maintains approximate knowledge of the system membership [13], and does does not involve the application in recovery after data is lost.

The important aspect of pbcast is a gossip-based repair mechanism, which assumes that each participant has addresses of some random subset of the participants. Each pbcast participant keeps a buffer of messages received recently (here, during the past second). At a constant rate (here, every 100ms) each participant picks some other participant and sends a unicast message describing the buffer contents. If a recipient of a gossip message is missing a message, it solicits a retransmission from the gossip source. Using a model developed in [2], the developer can tune the parameters that determine the frequency of gossip and the delay before messages are garbage collected to fit the setting.

Pbcast employs a mixture of IP multicast and unicast. Multicast is used to disseminate a new message and also to retransmit messages within regions where multi-participant loss is suspected (heuristics are used to detect this condition). Unicast is used to send gossip messages and to retransmit messages after individual loss. To isolate the impact of multicast retransmissions, we considered two versions of pbcast: a basic version that uses only unicast, and one we call pbcast-IPMC which employs multicast in the manner just described. We also considered one further extention to the protocol, called Pbcast-grb ("gossip about repair bits") in which processes gossip about which messages were recently retransmitted; Pbcast-grb behaves like Pbcast-IPMC, but also uses multicast when retransmitting a message for which, through gossip, a sender of the retransmission has learned that some other process previously unicast a retransmission of the same message. (An additional "hierarchical" extension to the protocol is introduced and described in Section D, when we present our experimental results for the Spinglass implementation of pbcast used in a local area network.)

C. SİMULATİON RESULTS

Simulations of simple network topologies

For our first NS-2 simulations, we constructed tree-topology networks with sizes ranging from 20 to 80 nodes. All of the trees have depth 4, and all nodes have an SRM or pbcast: protocol agent attached. The size of the process group in these simulations equals the size of the tree. There is one sender in the group, located at the root node of the tree; it generates 100 210-byte messages/second. We configure network links to have bandwidth of 1.5Mbits each, and simulate a low level of packet loss by having each link drop packets with probability 0.1%. It should be noted that no end-to-end data loss was observed: the reliability mechanisms of the protocols overcame these data losses.

For each group size and protocol, five distinct simulations were performed with different random seeds. Each simulation lasts 100 seconds during which 10000 messages are multicast to the group by the sender.

Our first analysis of overhead focuses on the retransmission "request" and "repair" messages received by each group member. Duplicate request and repair messages are taken into account in these measurements. Then, mean values are calculated for each simulation. These are not the only forms of overhead on the protocols: for SRM, we omit "session" messages from this measurement, while for pbcast, gossip messages are not included. Since both overheads are basically constant, we felt that mixing these forms of overhead with request or repair measurements would make the comparison confusing. Instead, as discussed momentarily, we include such costs in measuring link utilization.

The top two graphs in Figure 1 illustrate the results obtained in these first experiments. The x-axis is the group size and the y-axis is the rate of request and repair messages received per second, respectively. We see that, as the network and process group size scale up, the number of non-data protocol messages received by group members, as a result of recovery from data loss, increases linearly for SRM, an effect previously reported in [7,9,10,11]. These costs remain almost constant for Pbcast and Pbcast-IPMC. For the tree topology network simulations, adaptive SRM has a higher overhead compared to SRM with fixed timers, but later we will see that this depends on the network topology. Compared to the basic Pbcast protocol, Pbcast-IPMC has a slightly lower overhead in the form of request messages. Since Pbcast-IPMC multicasts repair messages for loss recovery in certain conditions, the repair message overhead increases relative to Pbcast. This is because some group members which did not actually request a retransmission, will nonetheless receiver a repair, or even multiple duplicate repair messages. On the other side of the coin, if a message was missed by multiple receivers, Pbcast-IPMC increases probability of rapid convergence.

The middle set of graphs in Figure 1 consider the link utilization close to the sender under the same conditions. To compute the utilization, we totalled the number of bytes on the link outgoing from the sender, and incoming to the sender, for messages of all types (including data and gossip). The units of the y-axis are percentage of the link bandwidth used by the test. For example, since this simulation involves sending 100 210-byte messages per second, or 168kbits/sec, the link utilization required just to send the data would be about 10%. Additional overhead results from retransmission requests, repairs, and gossip messages in the case of pbcast. We see that the link utilization rises rapidly as a function of group size for SRM in both its normal and its adaptive modes, while the utilization is lower for pbcast and also grows more slowly as a function of system size. We also ran a simulation of the same sort using a higher noise rate of 1.0% on each link, but obtained identical results; for brevity, we omit this data here. Notice that SRM is headed for trouble: at a group size of about 100 members, the sender's link will be saturated and packet loss will soar. Pbcast would apparently continue to function in much larger groups.

Our third set of graphs, at the bottom of Figure 1, repeats the measurements of overhead for pbcast and SRM but in a different network topology. Here, we organized the processes as a star with a single routing node at the center, and the

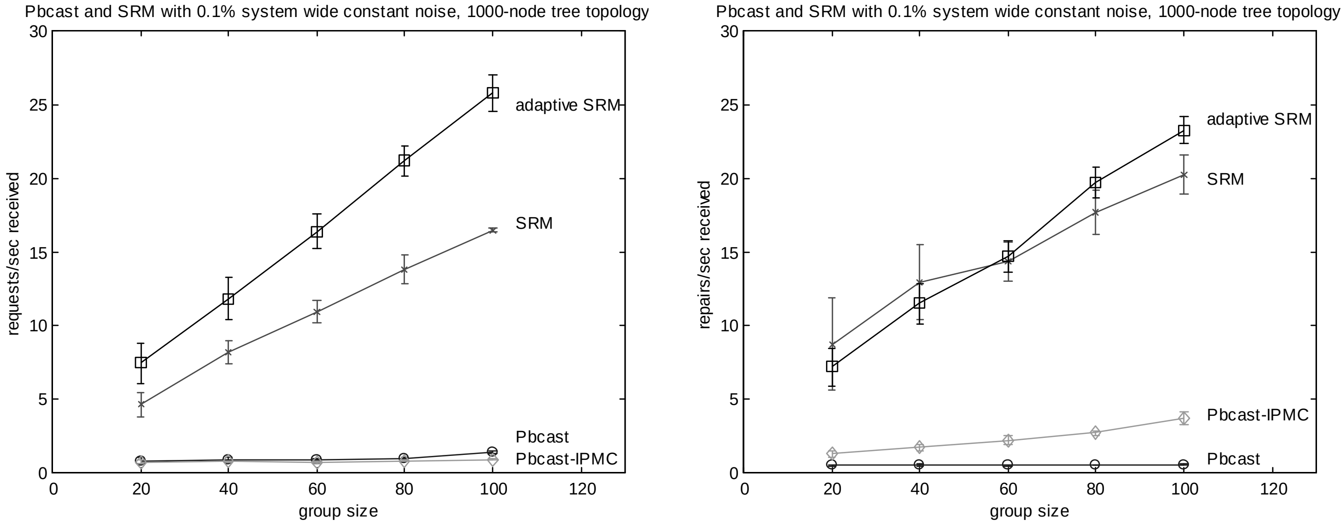

Figure 2: Request and repair rates in a 1000-node tree containing between 20 and 100 participants randomly scattered within the nodes of the tree. Error bars illustrate minimum and maximum values for 5 runs.

sender and receiver around the periphery. The normal SRM has a higher rate of repairs – mostly duplicates – but a lower rate of requests than the adaptive protocol. To understand what is happening, it is necessary to recognize that SRM uses probabilistic methods to avoid sending duplicate repairs: each process receiving a retransmission request has some probability of responding, with timing mechanisms used to inhibit duplicates. In doing this, SRM depends on the assumption that small numbers of participants are at any fixed distance from the sender. When all participants are equidistant, the inhibitory mechanism is defeated. Notice that the problem isn't necessarily unlikely: a star topology models what one sees in a local area network where most communication latencies are constant and small.

We also explored the impact of scaling the network to a very large size, while keeping the group itself at constant size. Figure 2 illustrates the results. Here, groups of various sizes were mapped onto a network of 1000 nodes (with tree depth set to 6, and a branching factor of 3) by randomly selecting nodes in the tree as group members. We set the message loss rate to 0.1% on each link, and ran five simulations with the sender injecting 100 210-byte messages per second. We then plotted the request and retransmission (repair) rates for the protocols. To give a sense of the variability of these results, we included error bars showing minimum and maximum values recorded over a set of five runs, using different seeds for the random number generator.

The data is consistent with our findings for the dense tree topologies used in Figure 1, although the SRM overhead values are somewhat higher. For example, in the 80-member case, the normal SRM request and repair rates rise to about 12 and 18/second respectively, double what we saw in a dense tree with the same number of group members. Similarly, the adaptive SRM protocol now has overheads of about 20 requests and 20 repairs/second, compared to 12 and 10, respectively, in the 80-member dense case. The higher rates are presumably triggered by the higher overall loss experienced as messages flow through the tree, since each link has an independent loss behavior. It is interesting to note that the rate of request messages seen by an average participant is about double that of the repair rate; one might have imagined that in SRM, each request would trigger a single repair. A reasonable inference is that the mechanisms by which SRM inhibits duplicate requests are not entirely effective here, so that roughly half of these requests are duplicates. Both versions of pbcast continue to have low costs; as in the dense case, the impact of multicast retransmissions is evident in a slightly higher rate of repairs.

Figure 3 considers inter-arrival distributions (throughput stability) of messages received by a typical participant for various sizes of network using the tree topology. Stable throughput is not normally considered to be a critical requirement in reliable multicast protocols, but we believe that there may be a substantial number of applications for which such a guarantee is important [2].

We ask several questions about these distributions. One concerns the absolute numbers: is a protocol quick to deliver messages? Obviously, this depends in part on the guarantees provided. Whereas SRM provides unordered delivery, the pbcast protocols provide fifo ordering -- messages are delivered in the order they were sent. When data is lost, subsequent messages are delayed until the missing data is recovered (the protocol gives up after a sufficiently long delay, but as noted above, all loses were repaired successfully in the tests reported here). Accordingly, pbcast interarrival spacings and latencies look like a superimposition of two curves, one for messages received with no loss, and a second for messages impacted by a loss (perhaps of some other message) and the delay until that loss was repaired. If we want latencies to be steady, we would hope that the distribution is unaffected by minor disruptions of the network, and that the primary impact of scale is that the absolute latencies grow, reflecting the larger average distance from the sender to the participants.

Figure 3 shows several such studies. The top two graphs show dense tree topologies (every node is a group member),

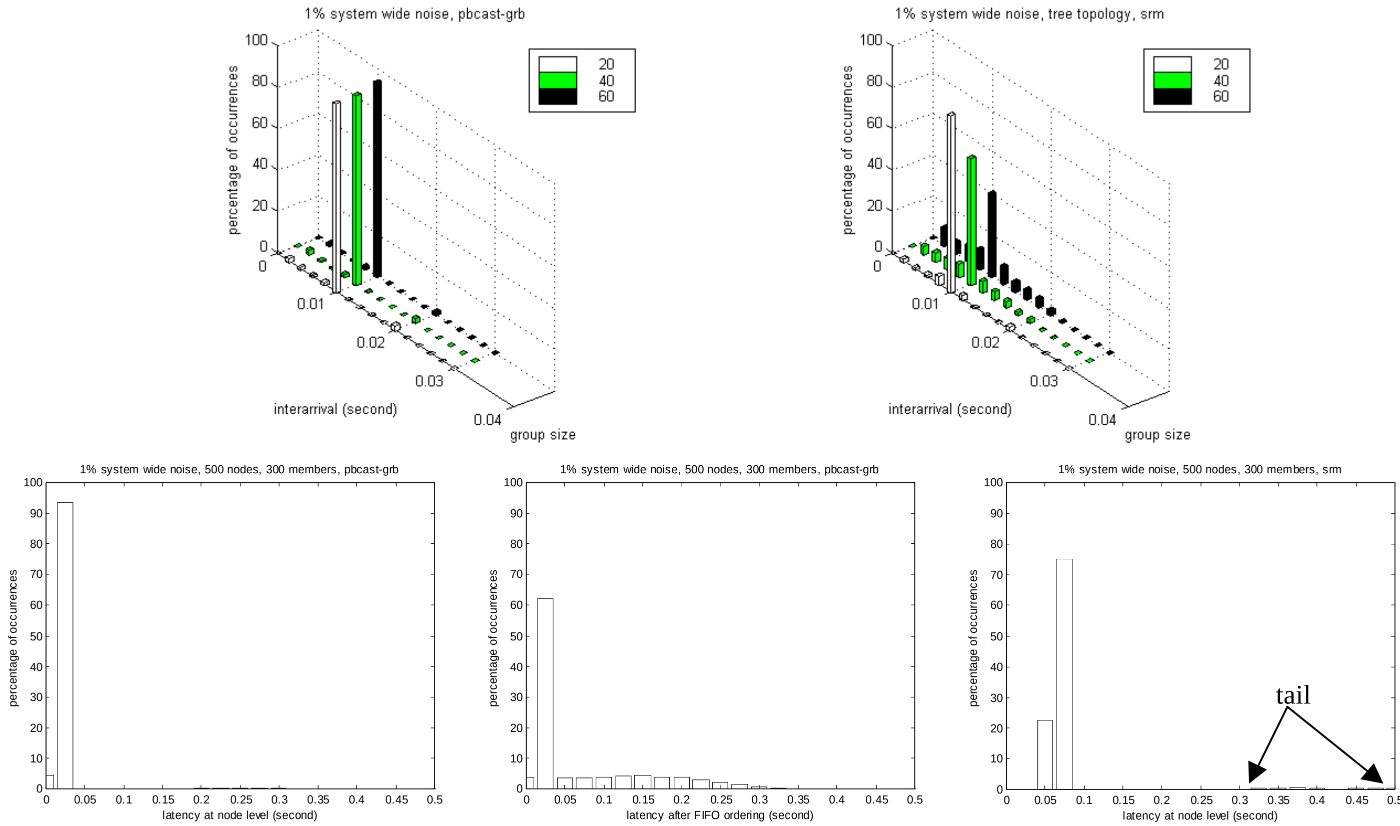

Figure 3: Top: Histograms of interarrival times pbcast-grb and SRM with 1% noise on all links in densely populated tree networks of various sizes. Bottom: The histograms on the left and middle show that even very small numbers of outliers can impact delivery latency if fifo ordering is desired in the case of pbcast-grb. The right-most historgraph shows latencies for SRM, an unordered protocol.

with 1.0% noise on each link, and data injected at the rate of 100 210-byte messages per second. On the left, we show the distribution of message inter-arrival spacings at a typical receiver for pbcast-grb; on the right for SRM.

We see that as the network size (and group size) is scaled, the distribution of pbcast-grb latencies remains essentially unchanged. As SRM is scaled, however, the distribution of SRM latencies changes as the size of the tree grows. The effect is fairly small, but the percentage of high-latency packets is clearly growing as we scale to larger groups: while both protocols have some high latency packets, SRM has a somewhat larger rate of high latency messages than does pbcast under the same conditions, and the actual latencies involved are larger by a factor of two or three. We will see additional evidence later suggesting that this phenomenon is real and can be quite severe. The implication is that SRM's recovery mechanism is somewhat slower to recover from lost messages than that of pbcast. The bottom set of three graphs in Figure 3 was based on a larger tree containing 500 nodes, 300 of which are group members, still with a 1% noise level. Here, we show pbcast without fifo ordering (left), with fifo ordering (middle), and SRM (right).

Although the scale of the graph may make this difficult to see, SRM has a large tail, with a maximum observed latency of nearly 500ms, and a group of packets delivered at around 400ms. Overall, SRM has a significant number of packets delivered during the first 100ms and a second broad distribution containing almost 6% of packets, which arrive with latencies between 300ms and 500ms. Notice also that the basic SRM distribution is not as tight as the unordered pbcast distribution, which has 90% of its packets arriving at the lowest possible latencies. In the case of pbcast, about 2.5% of packets are delayed and arrive in the period between 200ms and 300ms, with no larger latencies observed.

With fifo ordering, the pbcast distribution spreads, reflecting the cost of with waiting for messages to be retransmitted and placing them into the correct delivery order. Obviously if an application were to superimpose fifo ordering on SRM, a similar spreading of the delivery distribution would occur. However, because the maximum SRM latencies are quite a bit larger than those for pbcast, the resulting distribution would be broader than for pbcast, and would include a much larger percentage of the overall packet stream.

We believe these graphs to be important, at least in settings where steady delivery of data is required by the application. What we see here is that as SRM is scaled to larger groups, steadiness of throughput can be expected to degrade if the network drops some percentage of the packets. We experimented with a variety of noise levels, and obtained similar results, although the actual numbers of delayed packets obviously depends upon the level of noise in the system.

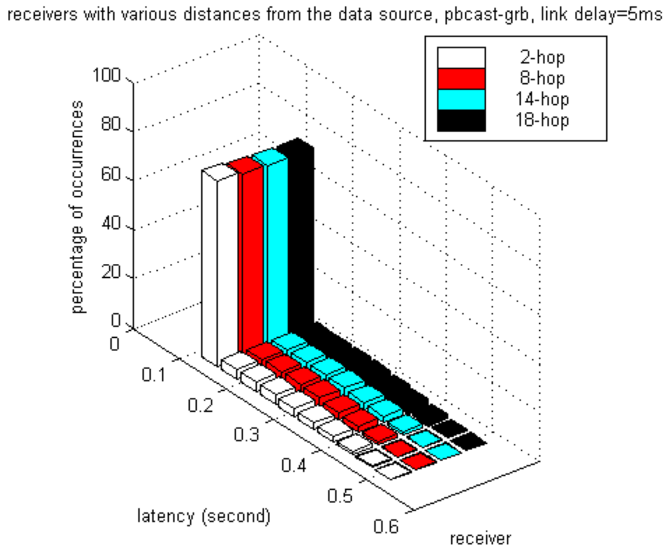

Figure 4: Using pbcast-grb, latency distributions are stable for receivers at various distances from the source.

Figure 4 plots latency distributions for pbcast-grb simulation in a 20-node linear (chain) topology. The sender injects 100 msgs/sec; its outgoing link has a 1% loss rate. Other links are lossless; each introduces a 5ms delay. Our interest is in the impact of distance from the sender on latency. Theoretical analysis of pbcast [2] suggests that the distribution shouldn't change, and this is confirmed by our simulation. The only effect is to introduce a small "offset" to the distribution, corresponding closely to the network delay itself. We obtain the same results in other networks, although the effect is particularly "clean" in this very simple case.

Experimental Results

We now shift gears and present some experimental results for an implementation of pbcast-IPMC. We have worked with several implementations of the pbcast protocols at Cornell, but the one discussed here is part of a new system which we call "Spinglass", in reference to a form of glassy materials in which local spins of component molecules contribute to exotic global properties. Spinglass is designed to operate as a free-standing protocol library for use in conventional local area networks. Over time, we plan to scale the system up until it can be used in very large networks containing collections of local-area networks interconnected by wide-area links. This goal motivates some of the questions we'll ask about Spinglass and pbcast in the next Section.

In Figure 5, we focused on the impact of the IP multicast retransmission feature of the pbcast protocol. We study protocol performance in a network of 35 Sun workstations interconnected by a 10MBit ethernet, with Spinglass configured to run one gossip round every 100ms. In a minor extention to the protocol described earlier and simulated, garbage collection in Spinglass employs an adaptive mechanism; in this experiment, messages were garbage collected after about 10 rounds (1 second). We transmitted 100 1k-byte messages per second, and examined throughput at a receiver. As seen on the left, we then perturbed the system by triggering a type of failure: periodically, we intercepted 10 successive outgoing IP multicasts at the data source and converted them into unicasts to a single randomly selected participant. These 20 multicasts are thus known initially at 2 out of 35 processes (it should be noted, however, that we get very similar results if we discard the multicast entirely, or unicast to more than a single process).

For the left-most graph, we disabled the IP multicast retransmission feature of the protocol. As might be expected, the disrupted multicasts are associated with bursts of gossip-based retransmission activity, disrupting throughput for a short period until all receivers have repaired their data streams. On the right, we see the same experiment but now we reenabled IP multicast when retransmitting messages which the protocol detects as having been dropped by multiple receivers (by seeing multiple retranmission requests). Small fluctuations remain, but the overall data rate is extremely steady. This makes sense if one considers the tight distributions of throughput predicted by our simulation for similar cases: the protocol on the left is basically the "pbcast" protocol while the one on the right is essentially the one called "pbcast-grb" in our simulations.

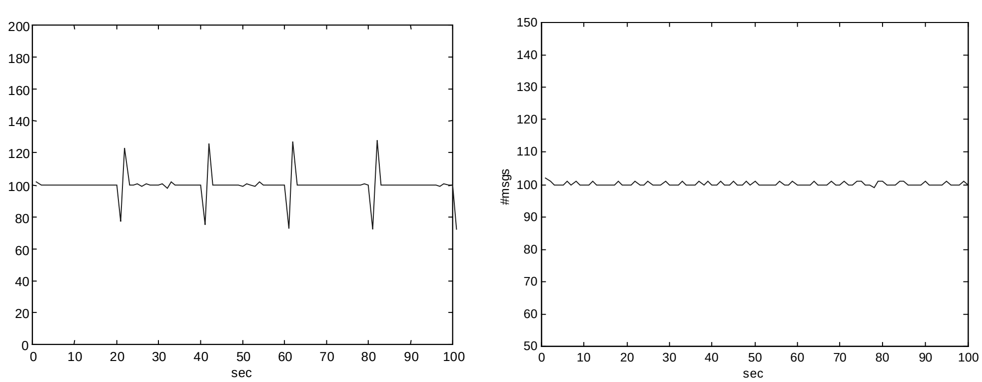

Figure 5: Received data rates (msgs/sec) for the Spinglass system without (left) and with (right) multicast retransmissions in a group of 35 processes with injected noise.

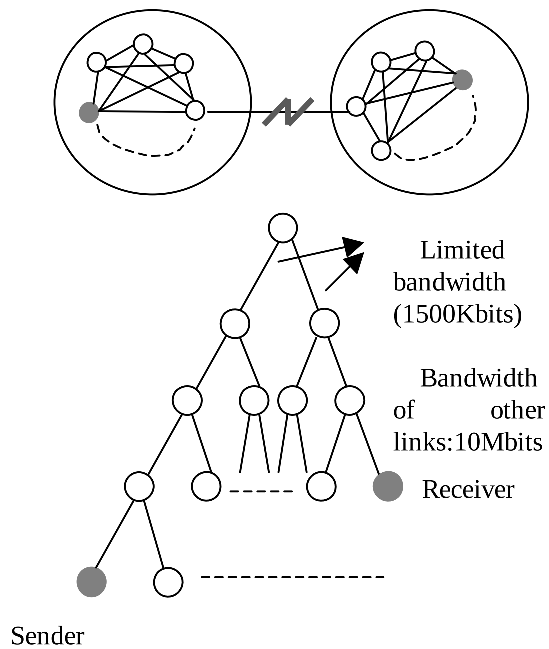

Figure 6: Top: A LAN with two clusters connected by a noisy link. Bottom: A tree with a router with limited bandwidth at root

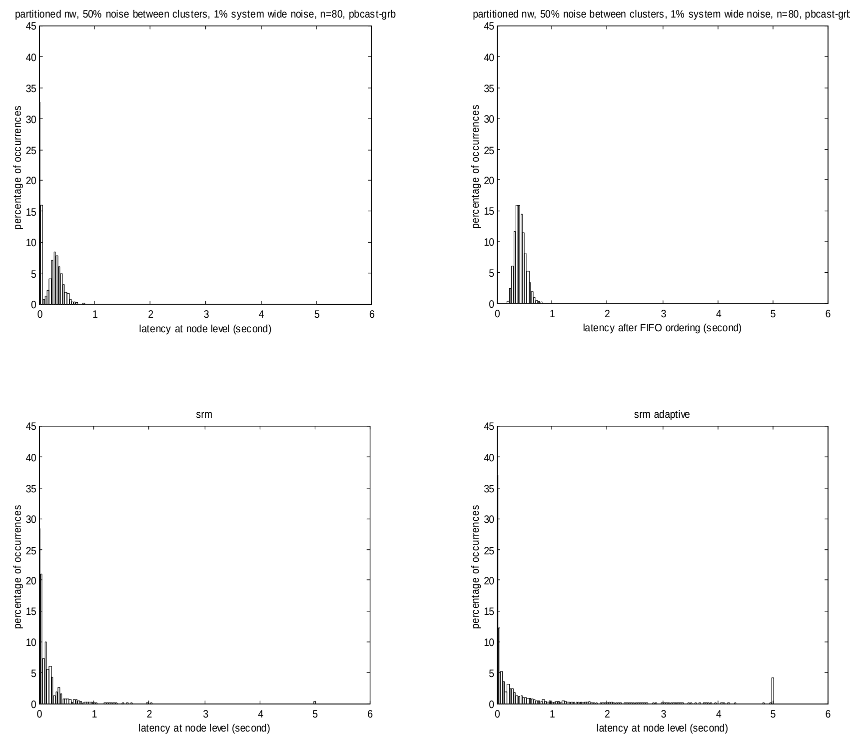

Figure 7: Delivery latencies in a two-cluster LAN.

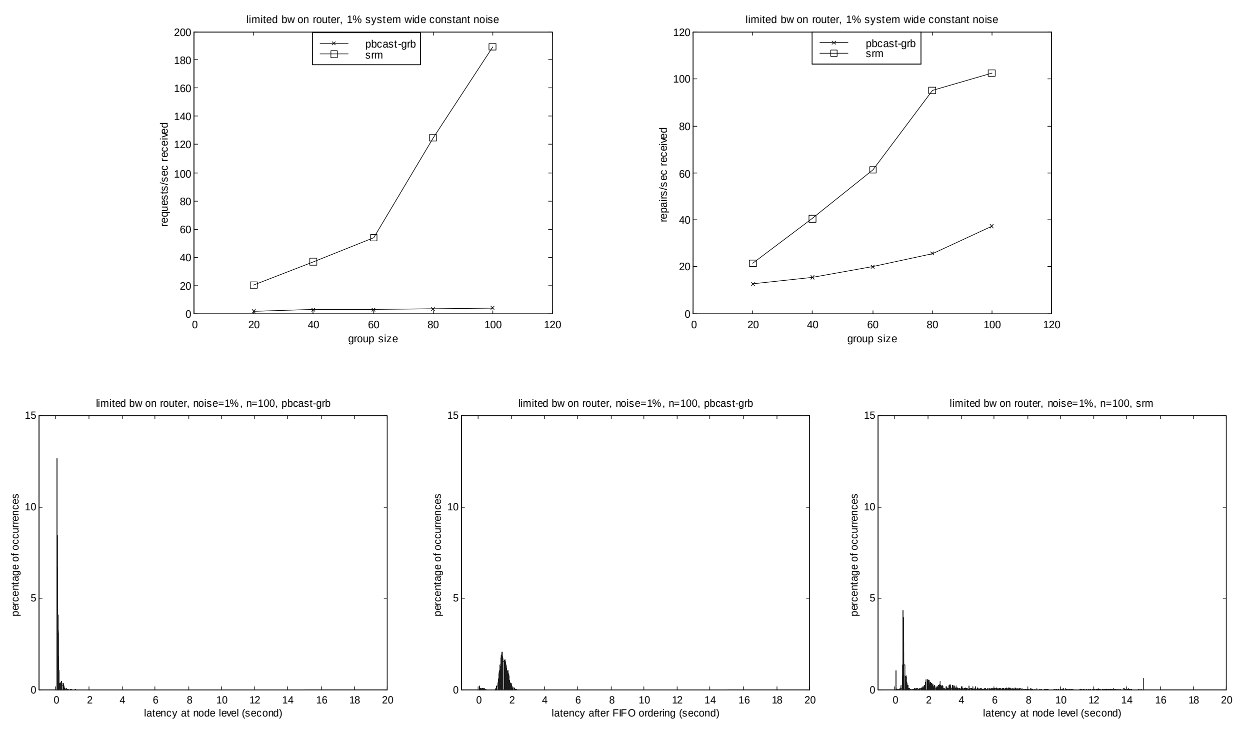

Figure 8: Top: Requests and repairs in a tree with limited bandiwth on root links. Bottom: Delivery latencies in such a tree.

D. SİMULATİON OF “CLUSTERED” NETWORK TOPOLOGİES

So far, our discussion have focused on the impact of randomized packet loss on the performance of the two reliable multicast protocols. However, there is an important issue not captured by such studies: how do these protocols perform in networks where routers with limited bandwidth connect groups of participants, as might occur if the protocol was used in an enterprise containing two local area networks linked by a longer distance link. Such configurations are common even in organizations of modest size.

The pbcast protocol might be expected to degrade dramatically in such a network, because of its gossip communication pattern. The problem is that unconstrained gossip presents the router with a load that rises linearly in the number of participants, because a significant number of gossip paths pass through the central router – the root of the tree. In a balanced tree, half of all gossip will pass through the root.

Figure 6 illustrates the two "clustered" network configurations for which we used simulation to explore the behavior of the protocol. At the top is a two-cluster dense 80-member configuration, in which two LAN's (modelled as fully connected networks) are connected by a single link. To emphasize the impact of noise on this link, we introduced 1.0% data loss on links within the clusters, but set the rate to 50% on the inter-cluster link. Below we see a second case of interest: a tree topology, in which the bandwidth through the links to the root node was limited.

Figure 7 illustrates our simulation findings for the two-cluster case. The sender is in one cluster, and we study latency at a receiver in the other. At the top, we see pbcast without fifo ordering and pbcast-grb after fifo ordering. The latency distribution remains relatively tight, in the range between 0 and 1000ms. Unlike the distributions seen in the tree structured network examined in figure 3, however, most packets are now affected by a delay; presumably, this is due to the high loss rate we imposed on the central link. SRM latencies, however, now exhibit a very long tail, particularly in the case of the adaptive SRM, which has a significant number of very delayed packets. It should be commented that the "spike" seen in the adaptive SRM data at latency equal to 5 seconds occurs because all packets with latencies greater than or equal to 5 seconds are counted in this single "bin." Thus, in this configuration, both SRM and adaptive SRM deliver some packets with very long delays of many seconds, particularly in the adaptive case, and roughly 5% of all packets are delayed by 5 seconds or more before delivery.

For two-cluster networks in which stable latency is desired, the adaptive SRM protocol would clearly not be a suitable candidate. The normal SRM protocol does better, but still has a significant percentage of packets arriving with latencies in excess of 1 second. Pbcast, on the other hand, delivers all data within 1 second and hence can be seen as offering relatively steady data throughput in networks with this configuration.

Turning to our second example, namely the network shown in the lower half of Figure 6, we limited the bandwidth near the router to 1.5Mbits per second and experimented with trees of size 20, 40, 60, 80 and 100 nodes, where each node is a group member. As seen in the figure, one of the nodes is designated as the sender; it initiates 100 1000-byte multicasts per second (800Kbits/second – about half the capacity of the root node). All other nodes are receivers, but we measured the reception rate at a receiver far from the sender, as shown. The data loss rate was set to 1.0% for the entire network.

Consider first the request and repair rate graphs in Figure 8. For 20 and 40-node simulations, we see curves very similar to the ones seen early in this paper, in Figure 1. However, as the group size grows, the rate of growth of requests and repairs rises very sharply, reaching a rate of almost 200 requests/second in the 100-member group, and 100 repairs/second. When one estimates the bandwidth consumed by the repair requests, the reasons for this are clear: the router is becoming saturated and the loss rate near the root is rising as this occurs.

The rate of requests seen in the case of pbcast remains nearly constant in this test, and the growth in repairs seems consistent with the size of the group and the higher noise rate used in this experiment. In fact, pbcast-grb is headed for trouble here too – the central router becomes saturated when the total of requests plus repairs approaches 100 – but we can see from this graph that the group would need to become quite large before such an effect might arise.

Turning to the latency histograms, the picture is consistent with these findings. Shown are latencies for pbcast-grb with and without fifo ordering, and for SRM, all in the case of a 100-node group. Latencies for the two pbcast protocols are consistent with what we have seen previously; the long distance from sender to receiver accounts for the "offset" evident in the fifo case. The problem here is that when a message is lost close to the router, the protocol pays a fairly high cost to recover it; this concerned us enough to explore possible remedies, as discussed below. But the picture for SRM is grim. Here, the latency distribution has become extremely spread out, with most messages delayed for more than 2 seconds, and a significant percentage delayed for 15 seconds or more – again, this graph collects all data points greater than a maximum value into a single bin, at 15-seconds in this case. Considering that the actual latency from sender to destination was on the order of 75ms in this graph, such enormous delays point to a real breakdown in the behavior of the protocol.

The three graphs shown in Figure 9 use Spinglass to explore the two-cluster problem from a practical perspective. On the far left we see a graph of load measured at a router. The axis shows the number of messages per second passing through the router, and the fluctuations reflect the normal variations to be expected in light of the random nature of gossip algorithms. As noted earlier, about half of all gossip messages will pass through the central link in a balanced two-cluster network. Our experimental setup here is similar to the one described in the previous section, but now we use a configuration with 26 machines, because this was the largest two-cluster configuration of similar computers available to us. The 26 machines split into one cluster containing 15 machines and one containing 11. Since 11 is a bit less than half of 26, we would expect that a little less than half the gossip messages pass through the central router.

In this experiment, we send 100 messages per second, and each process gossips once every 100ms – 10 times per second. No noise was injected. Under these conditions, 260 gossip messages per second are produced, plus additional overhead associated with retransmission requests and repairs. Our graph shows only protocol messages, so we see the residual

140 messages/second here – indeed, roughly half. The midle graph shows how these rates vary with group size: not surprisingly, the normal load linear is in the size of the group.

This pattern of gossip suggests that in a large configuration, the central router of a large system would become overloaded. Moreover, if one reflects upon the nature of this gossip, it becomes clear that the "useful content" of the messages should be relatively constant: at a given point in time, gossip from one densely connected cluster to another should be statistically similar. Thus, while the volume of messages flowing from one side to the other rises linearly with group size, the information content of the messages can be understood as dropping at a similar rate. To remedy the problem, we implemented a hierarchical gossip scheme, using an idea first suggested by our colleague Robbert van Renesse, who did something similar in his work on a gossip implementation of a distributed management information base. The intent of van Renesse's method is that each level of a network should see constant gossip load, and that the information content of gossip messages should be as high as possible.

The basic insight underlying hierarchical gossip is that in a tree-structured network containing an evenly distributed group of *n* members, if each gossips with equal probability to all the others, a central node will see about half the messages. A level 2 node will see about a fourth of them, etc. The hierarchical gossip mechanism works to compensate for this by having each group member compute a *weight* for each other group member based on the inverses of these percentages, and then to employ these weights to adjust the probability of gossiping to that member. In the case where all weights are equal, we will have the normal gossip protocol.

To illustrate a hierarchical weighting, suppose that the group has total weight *w* and splits into two clusters of equal size. Now, suppose that each member on the right side of a balanced tree allocates this weight so that the total weight for members on the left side of the tree is *w/4* while the weight on is side is *3w/4.* The effect will be to reduce the frequency of messages from that member to the left side of the tree by half relative to what would occur if each side had weight *w/2* from the perspective of this member. Notice that each member would independently compute this weighting. In Spinglass, we do so by examining IP addresses: we think of the IP addresses as if they defined a tree, and assign weights using this tree; since each member has approximate knowledge of the total membership of the application, this is simple to compute and entirely local to each member.

In general, it can be shown that an exponential weighting gives the optimal pattern of gossip: the weighting should be proportional to $1/k^d$, where *k* is the branching factor of the tree, and *d* is the distance of the target member from the member computing the weight factors. It is not hard to show that this approach will indeed produce constant load at central routers.

As seen by the bottom curve in the left-most graph, the direct impact of hierarchical gossip is to reduce the actual traffic through the central link of our experimental system. For this particular setup, the hierarchical scheme reduced the load on the router to 20 messages/second. As seen in the middle graph, the load is now constant as the group size increases.

However, the right-most graph points to a problem. This graphs propagation delay of multicasts (latency). As we see, when moving from the normal method to the hierarchical one, the delay increases significantly.

Fortunately, a simple remedy eliminates the problem. The "fast hierarchical" scheme is obtained by decreasing the gossip round length to increase the frequency of gossip, while keeping the hierarchical weighting unchanged. Now latencies drop below those of the original protocol, yet the load on the router is only slightly increased.

In very large networks, we may move to a simpler weighting scheme. Our thinking is to maintain reasonably accurate lists of the membership of the cluster to which each member belongs, but very coarse information about remote clusters. We would then use a form of hierarchical gossip to control rates of gossip in these two cases. However, this extension of the hierarchical gossip method is well beyond the scope of our present study, and will be reported elsewhere.

E. DISCUSSION

Our investigations yield some general conclusions about the behavior of scalable reliable multicast protocols in large systems, and some specific conclusions about the relative advantages and disadvantages of SRM vis-à-vis pbcast, and about the limitations associated with each protocol.

Limitations of the SRM protocol

Our work points to a number of limitations of the SRM

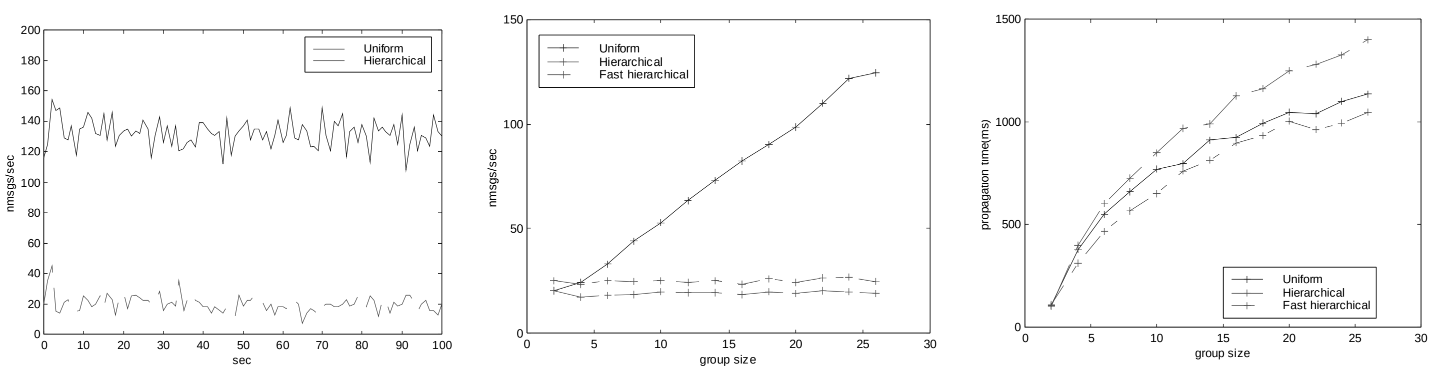

Figure 9: Experiments with a hierarchical modification of pbcast implemented in Spinglass.

protocol, some more serious than others. Overall, we see that when a network has lossy links, even if the loss rate is low, SRM can generate very high rates of overhead, an effect first observed in [7,9,10,11]. This takes the form of requests for retransmissions of data, sent using multicast and hence seen by significant numbers of processes, and repair messages, also sent using multicast. Although our tests used the best available SRM implementation, parameterized in the manner recommended by the developers, and considered a variety of network topologies, the SRM timing mechanisms and inhibition mechanisms prove to be only probabilistically stable under these conditions. A consequence is that as the network grows larger, the absolute rate of "mistakes" increases. These take the form of duplicate requests and duplicate repairs, and of requests or repairs transmitted with inappropriate network granularity (TTL values), and are manifest as excess traffic into typical participants.

One could argue that although undesireable, these overhead messages are still a relatively minor phenomenon in networks of the size investigated here. The growth rate predicts that routers may eventually become saturated, but one can question whether it is realistic to consider a loss rate as high as 1.0%, and lower data loss rates obviously would reduce overhead. Furthermore, the impact of overhead is relatively minor if the absolute data rates turn out to be sufficiently small compared to the capacity of the network.

A more serious problem is evident in the latency distributions for SRM in two-cluster networks with a noisy connecting link. This, we believe, is not such an unlikely situation – and while our loss rates may be on the high side, it is not unlikely that loss rates would at least sometimes reach these levels in typical LAN configurations or networks with a WAN link between two LANs. Under such conditions, we saw that delivery latency for SRM can soar to thousands of times the actual source-to-destination network latency, with worst-case figures of 15 second or more in one test, compared with 75ms source-to-destination network latencies. A significant percentage of SRM packets experience long delays, and many applications would thus be forced to buffer very large amounts of data. For applications in which data freshness is at all important, this would seem to be a real drawback for the protocol.

Limitations of the Pbcast protocol

Turning to the pbcast protocol, while the picture is somewhat more positive, we also see some problems. Overall, under the conditions where SRM shows severe overhead growth, pbcast sometimes shows moderate growth. Thus, pbcast also faces some scalability limits, although they seem unlikely to emerge until a network grows quite large. Most serious is the issue of gossip load on centralized links in multi-clustered networks. Here, we saw that unless something is done, the pbcast protocol will load such a link to a degree proportional to the size of the group. If the capacity of the router is exceeded, this will trigger a high rate of loss on the corresponding link; even if not, the behavior is unfortunate and would impact other applications sharing the network.

It is interesting to see that although such loss presumably occurs in our simulations of networks with limited bandwidth near the central router, the actual "quality" of the protocol is not much impacted. This makes sense, since gossip messages are relatively redundant, and at any point in time, gossip from one "side" of a network to the other might be expected to be statistically similar. Thus, this huge volume of gossip carries relatively little useful information.

With this insight in mind, the success of the hierarchical gossip strategy makes sense. Basically, we cut down the frequency of messages – so the average gossip message carries more useful content. The router experiences less load and imposes less delay on messages, and a small adjustment to the frequency of gossip rounds compensates for the (minor) growth in latency otherwise seen.

Comparisons

Overall, we believe that our studies show pbcast to be a better behaved reliable multicast protocol than SRM in the settings considered here. In fact, we developed pbcast and implemented Spinglass for a different reason: we sought a protocol offering a rigorously quantifiable form of reliability but weaker than virtual synchrony [3], the reliability model with which our team has worked in the past. Pbcast's bimodal reliability guarantee seemed to fit the bill. The extremely stable throughput of the protocol came as something of a surprise, but one we view as very useful in practical settings.

When we started our comparisons of pbcast with SRM we expected that the two protocols would, in all likelihood, offer similar behavior. Obviously, their repair mechanisms are very different, and we did expect to see some evidence of this in our experiments. It came as a surprise to us that SRM overheads were so high compared to those of pbcast, and that SRM latency distributions are so poor compared to those of pbcast.

Upon reflection, we believe that these findings are easier to understand. Basically, the issue seems to be one of how random low-probability events impact these kinds of protocols. SRM has a number of timers and inhibitory mechanisms which are parameterized for the specific network. We can view these as probabilistic mechanisms for overcoming data loss. By introducing *system-wide* data loss, even at a low rate (the 0.1% loss rate is, we believe, reasonably low) SRM's assumptions are apparently defeated as the network grows large.

This makes some intituitive sense. Suppose that processes *p* and *q* are receivers in an SRM session, and are symmetric in all respects: they are at the same distance from the sender, etc. The basic hypothesis of SRM is that most multicasts will be delivered reliably by IP multicast, and that the primary forms of loss are entirely local (*p* drops a message, but nobody else does so) or regional (a subtree drops a message, but no other subtree does so). SRM then repairs these problems locally.

In our tests, which could be criticized as unfair to SRM, *p* and *q* could both experience data loss – directly, or because packets are lost in both sides of a large spanning tree. In either case, the timer mechanisms for SRM are supposed to inhibit duplicate retransmission requests. But if we scale the network, *p* and *q* are further and further away from each other, and there are more and more processes in each set of mutually symmetric processes. The SRM mechanisms (even the adaptive ones) make no provision for this effect. Thus, *p* and *q* become more and more likely to both solicit a retransmission. By similar reasoning, it becomes more likely for multiple processes to respond to a single retransmission request. This was particularly evident when all participants were equidistant.

In real networks, SRM seems likely to behave better than our experiments might suggest, but it is clear that the actual situation will depend very much on the dynamics of the network and the types of packet loss experienced within it. Moreover, some SRM extensions (local repair, scalable session messages) were not available to us for simulation.

Nonetheless, we see a fundamental advantage to pbcast relative to SRM. Unlike SRM, pbcast makes only weak stochastic assumptions, which apply to all messages and all processes. The basic gossip mechanisms are highly randomized, and effectively random data loss is attacked by randomized gossip repair. The laws of probability now work in our favor, and the exponential convergence of gossip towards full diffusion of data in the network similarly benefits us. As long as IP multicast is used very sparingly for retransmission (and pbcast-IPMC and pbcast-grb are both very conservative protocols), the typical multicast starts by reaching most processes, and all that remains is a small amount of repair. The hierarchical pbcast mechanisms can be seen as analogous to SRM’s regional repair mechanisms.

F. PRIOR WORK

We believe that our study is the first to consider the specific questions raised here with the goal of understanding the stability of latency distributions in scalable reliable multicast setttings. However, we are certainly not the first to have criticized SRM, and some previous studies have proposed SRM-like protocols with better local repair strategies giving faster convergence with less overhead. First among this was work by one of the SRM developers, Liu, who also proposes some extensions to improve behavior of the protocol [10]. Lucas, in his thesis, identified similar issues with SRM [11].

Li and Cheriton have proposed a reliable multicast protocol called OTERS, which provides low recovery latency and low recovery traffic levels while requiring some additional network support [9]. Their work uses NS-2 to compare OTERS with SRM and TMTP, another well-known reliable multicast protocol. They simulate “transit-stub” network topologies with sizes 100 and 600, with groups of sizes 10 and 60 respectively, and link-error rates of 0.5%. The simulation study focuses on the analysis of recovery latency and traffic load for loss recovery. Among their findings, the authors note that SRM can perform poorly. However, they did not encounter anything as extreme as what we saw in our two-cluster simulations.

In a very recent study, Hanle and Hofmann use NS-2 to compare the performance of SRM, MFTP (Multicast File Transfer Protocol) and MFTP/EC (MFTP with Error Correction) [7]. The results are similar to our findings for SRM's link utilization. The paper concludes that SRM can easily encounter situations in which multiple repair packets are multicast in response to a single retransmission request. In general, there is a trade-off in SRM between duplicate packet flow and loss recovery speed.

G. CONCLUSION

We studied two reliable multicast protocols in a variety of conditions with attention to growth of overhead and distributions of latency. In networks with even very low levels of data loss on the average link, we found that the SRM protocol experiences faster growth in overhead than does pbcast, and in cases where there is long-haul link connecting two or more clusters of sites SRM shows a dramatic degradation of latency. Pbcast latency seems stable as a function of network size and group size and, in the tests reported here, overhead growth is very modest. The major problem identified for pbcast involves overloading of central routers, and can be addressed using a hierarchical scheme.

The Spinglass implementation of pbcast will be available, at no fee and in source form starting late in 1999.

H. ACKNOWLEDGEMENTS

We are grateful to Robbert van Renesse for his many suggestions and comments, and to Katie Guo, who helped us develop the NS-2 simulation of the pbcast protocol.

I. REFERENCES

[1] Sandeep Bajaj, Lee Breslau. Deborah Estrin, et.al, Improving Simulation for Network Research, USC Computer Science Dept. Technical Report 99-702, March 1999.

[2] Ken Birman, Mark Hayden, Oznur Ozkasap, Zhen Xiao, Mihai Budiu, Yaron Minsky. Bimodal Multicast. Submitted to ACM Trans. on Computing Systems. Also Cornell University CS TR 98-1667.

[3] Ken Birman. Building Secure and Reliable Network Applications. Manning Publishing Company and Prentice Hall, 1997.

[4] Brenda Baker and Robert Shostak. Gossip and telephones. Discrete Mathematics, June 1972.

[5] A. Demers et. al. Epidemic Algorithms for Replicated Data Management. Proceedings of the 6th Symposium on Principles of Distributed Computing. (Vancouver, CA; Aug. 1987) 1—12. Also Operating Systems Review 22:1 (Jan. 1988), 8—32.

[6] Sally Floyd, Van Jacobson, Steve McCanne, Ching-Gung Liu and Lixia Zhang. A reliable multicast framework for lightweight sessions and application-level framing. Proc. ACM SIGCOMM, 1995.

[7] Christoph Hanle, Markus Hofmann, 'Performance comparison of Reliable Multicast Protocols using the Network Simulator ns-2', Proceedings of the Annual Conference on Local Computer Networks, October 11-14, 1998, Boston, MA.

[8] Katie Guo. Scalable Message Stability Detection Protocols. Ph.D. thesis, Cornell University, May 1998.

[9] Dan Li, David R. Cheriton, 'OTERS (On-Tree Efficient Recovery using Subcasting): A Reliable Multicast Protocol', Proceedings of the 6th IEEE International Conference on Network Protocols (ICNP'98), October 1998, pp 237-245.

[10] Ching-Gung Liu. Error Recovery in Scalable Reliable Multicast. Ph.D. thesis, USC, December 1997.

[11] Matt Lucas. Efficient Data Distribution in Large-Scale Multicast Networks. Ph.D. thesis, U. Virginia, May 1998

[12] Boris Pittel. On spreading a rumor. In *SIAM Journal of Applied Mathematics,* Feb. 1987

[13] Robbert van Renesse, Yaron Minsky and Mark Hayden. A gossip-style failure detection service. *Proceedings of Middleware, 1998.*