LRRM: A Randomized Reliable Multicast Protocol for Optimizing Recovery Latency and Buffer Utilization

Nipoon Malhotra, Shrish Ranjan, Saurabh Bagchi
Dependable Computing Systems Laboratory, School of Electrical & Computer Engineering, Purdue University
E mail: *nipoon.malhotra@alumni.purdue.edu, {sranjan, sbagchi}@purdue.edu*

Abstract

An efficient recovery protocol for lost messages is crucial for supporting reliable multicasting. The tree-based recovery protocols group nodes into recovery regions and designate a recovery node per region for buffering and retransmitting lost messages. In these protocols, the recovery host may get overloaded during periods of large message losses and costly remote recovery may be initiated even though a peer node has the lost message. To address these drawbacks, the Randomized Reliable Multicast Protocol (RRMP) was proposed which distributes the responsibility of error recovery among all members in a group. The pressure on the buffer and computational resources on the intermediate nodes is increasing due to the wide distribution of multicast participants with widely varying reception rates and periodic disconnections. In this paper, we propose the Lightweight Randomized Reliable Multicast (LRRM) protocol that optimizes the amount of buffer space by providing an efficient mechanism based on best-effort multicast for retrieving a lost message. A theoretical analysis and a simulation based study of two realistic topologies indicate that LRRM provides comparable recovery latency to RRMP for lower buffer space usage. While presented in the context of RRMP, LRRM can also benefit other tree-based reliable multicast protocols.

Keywords: *Reliable multicast, Randomized protocols, Buffer utilization, Recovery latency, Tree-based multicast protocols.*

1. Introduction

Multicasting is an efficient way of distributing data from a sender to multiple receivers. IP multicasting uses best effort delivery semantics which implies that multicast packets can be lost, delayed, duplicated or delivered out of order. There has been considerable interest in augmenting the best-effort nature of IP multicast protocols to support reliable multicast capable of tolerating message losses and node failures. A number of solutions have been proposed, notable among which are the tree based protocols. The basic tree-based protocols (RMTP [4], TRAM [5], TMTP [11] and LBRRM [9]) group the receivers into a number of regions based on criterion such as administrative domains, geographical proximity, or distance from the sender. Each region selects a designated recovery host to facilitate recovery of lost packets in that region. The designated recovery host buffers packets to enable local recovery of the packets. These protocols, however, suffer from the drawback that under high message loss rates in a region, the designated recovery host may get overloaded with recovery requests. Further, costly remote recovery may be performed even if a host in the local region has the packet being requested since packets are only requested from ancestors in the tree structure and not from siblings. Birman *et al.* proposed the Randomized Reliable Multicast Protocol (RRMP) [1] to address these drawbacks. RRMP improves the robustness of tree-based protocols by diffusing the responsibility of error recovery among all members in a group. RRMP groups receivers into a hierarchy, similar to the tree-based protocols. However, RRMP lets each receiver, which experiences a message loss, send its repair request to a randomly selected receiver in its local region and with a small probability, to some randomly selected receiver in a remote region. If no reply is forthcoming from the initially selected node, then the receiver picks a different node to try the local and the remote recovery. The reliability of the protocol depends on statistical properties of its randomization algorithm which have been formally analyzed in [1]. These parameters, such as the probability of initiating a remote recovery procedure, can be tuned according to application requirements for message traffic and delay. The results demonstrate that the protocol achieves fast error recovery with low overhead, compared to tree-based protocols like TRAM [5]. In [2], the authors propose techniques to reduce the buffer requirements at

the intermediate hosts that may act as recovery hosts and show the tradeoff between recovery latency and buffer requirement.

As reliable multicast protocols are deployed over wide area networks, it is a likely scenario that the intermediate nodes are light weight and constrained in their buffer space and processing capabilities. Further, receivers with widely varying reception rates and periods of disconnection result in large buffer space requirements. The recovery nodes may be network devices, such as switches, that service multiple flows. Therefore it is important to minimize the amount of buffer space used at intermediate recovery nodes. In this paper, we propose a new protocol, based on RRMP, called *Light-weight Randomized Reliable Multicast* (*LRRM*) to solve this problem. LRRM, like RRMP, distributes the responsibility for recovery among multiple nodes in a region. However, it uses an efficient multicast based method for disseminating the request for locating the recovery host to retrieve lost packets. This allows the protocol to reduce the number of redundant copies of a packet that need to be buffered in a region without affecting the lost message recovery latency. Overall, LRRM shows better storage space utilization, i.e., exhibits lower buffer usage at intermediate nodes for similar recovery latency. However, LRRM suffers when the network configuration is very dynamic and nodes either move or die. We show the cross-over point for LRRM with respect to RRMP. From a theoretical analysis, we see that for an environment which is not extremely dynamic in terms of node joins and leaves, LRRM has lower recovery latency compared to RRMP. For example, in a group of size 20, if the group membership changes less frequently than every 30 lost messages, LRRM is favored. For more frequent group changes, the cost of multicast tree formation in LRRM causes it to under-perform RRMP. The theoretical analysis also brings out the reduction in buffer requirement in LRRM. For example, it shows that for RRMP, at least one-third of the nodes in a region need to buffer packets to achieve a recovery latency equal to that of LRRM in which at most one node per region buffers a packet.

We build simulation models for RRMP and LRRM in ns-2 [6] and perform simulations of two topologies. The first is a real-world topology closely based on the SURAnet backbone, a contemporary WAN interconnecting research institutions in the Southeastern US. The second is a transit-stub topology of 50 nodes generated using the Georgia Tech Internet Topology generator (GT-ITM). The results for topology A show buffer requirement reduction of 10-25% and that for topology B, 100-125%. The simulation results also show that the buffer requirement in LRRM, to maintain a given recovery latency, scales down gracefully as the message loss rate in the environment decreases.

The rest of the paper is organized as follows. Section 2 describes related work. Section 3 describes the RRMP protocol, the motivation for our work, and presents the LRRM protocol. Section 4 describes the theoretical analysis of LRRM and RRMP. Section 5 presents the simulation setup and the experiments. Section 6 concludes the paper and provides some directions for future work.

2. Related work

Reliable multicasting is a well researched problem. Notable among the proposed solutions are deterministic tree based protocols, randomized tree based protocols, and probabilistic protocols.

Deterministic tree based protocols have been shown to be a scalable way of reliably disseminating multicast messages that avoid the problem of ack/nack implosion at the sender. Several deterministic tree based protocols have been proposed, such as RMTP [4], TRAM [5], LBRRM [9], and TMTP [11]. These protocols are based on the general principle of forming repair groups and logically arranging them in tree-like hierarchies. Each repair group has a repair head which caches messages from the sender and services requests for lost messages from the rest of the group members. In case a message is missing at the repair head, it is retrieved from the repair head of the next higher layer in the tree. For dynamic tree based protocols, (e.g., TRAM) the repair head may be reselected based on changing network topologies. Having one such designated repair head has many important implications. First, it is the responsibility of the repair head to maintain a cache of all the messages transmitted to any node in the group. A majority of message recovery requests from the members in a local region should be serviced by the messages in its repair head's cache. This scheme of having a single repair head per region makes the administration of such regions tractable. All members of the region need to know only the address of this repair head. However, the repair head becomes a possible single point of failure. During periods of large message losses in a region, the repair head may be overloaded with requests for recovery and therefore become a performance bottleneck.

Randomized tree based protocols work by distributing the burden of message recovery among all the members of the local region. This strategy known as *distributed error recovery* is the basis for a number of protocols including RRMP. The reliability of such protocols depends upon the statistical properties of the

underlying algorithms. Bimodal multicast [12] is a protocol for multicast from many sources to many destinations. In this protocol, a node *p* exchanges its message history with a randomly selected neighbor *q*. The node *q* asks *p* for any message that it has not seen in an effort to converge to identical message history suffixes. After a certain length of time if a message cannot be recovered, the protocol declares failure and reports it to the higher application layer. It has been shown that the protocol has a bimodal property – with a very high probability, all the nodes will get the multicast message, a small probability that very few nodes will get the message, and a vanishingly small probability that most but not all the nodes will get the message. The recovery latency for the protocol is arbitrary depending on the tunable parameter of periodicity of message exchanges. Another important protocol belonging to this category is Scalable Reliable Multicast (SRM) [14]. In SRM, when a member detects a message loss, it initiates a recovery procedure by multicasting a retransmission request in the local region. Any member having the desired message in its cache responds by *multicasting* the message with a back off mechanism being used to prevent redundant requests and replies. Since the replies are multicast, in the cases of isolated members losing a message, SRM leads to redundant message traffic. The hierarchy in SRM is used only for distributing session messages. SRM does not exploit the tree structure to designate local recovery regions for sending retransmission requests and replies. Since SRM is designed for "many-to-many" multicast applications, the hierarchy is not organized with respect to a given source.

Probabilistic reliable multicasting protocols are based on the concept of gossiping and have been proposed mainly for environments where the underlying network provides little determinism, such as sensor networks. Anonymous Gossip (AG) [15] is one of the early protocols to employ probabilistic techniques to provide reliable multicast. In this protocol a node randomly contacts another member of the multicast group and sends it information about messages that the node has received or not received. The receiver checks to see if it has received any of the messages listed and based on this, the two nodes exchange messages which are not part of each others message history. However, AG shifts the responsibility for membership management and initial packet dissemination to the Multicast Ad Hoc On-Demand Distance Vector (MAODV) layer and thus presupposes the existence of a multicast infrastructure, albeit best-effort. Route Driven Gossip (RDG) [10] is another protocol that falls in this class. It provides reliable multicast services on top of best-effort unicast. Each member of a multicast group "talks" to a random subset of members of the group about its knowledge of the "state" of the group, e.g., the multicast messages received by the group. Lost state is recovered in a peer-based style. Each member of the group periodically sends gossip messages with IDs of recent received and missing messages to a random subset of members in the group to whom the routes are known. A given receiver responds to this "gossiper-pull" by sending the requested message only if the requested message won't be gossiped by it anymore. Stochastic analysis using epidemic theory enables the protocol to achieve a desired tradeoff between reliability and scalability by adjusting the protocol parameters and provide performance prediction of such protocols. RDG uses the parameters of *fanout*, the number of the randomly chosen members for gossiping, and *quiescence threshold*, maximum number of times a message can be gossiped, to obtain the required levels of reliability. These protocols differ from the randomized tree-based protocols in that they do not assume or create a tree structure in the network. These protocols are targeted to environments either with very large scalability requirements, e.g. hundreds of thousands of nodes as in a sensor network, or large error rates, e.g. ad-hoc wireless networks, where the only feasible reliability guarantees are probabilistic. In our environment, we have to provide deterministically reliable multicast and therefore the design point is different.

3. Protocol Description

In this section, we present the details of RRMP, the motivation for our modifications, and then our protocol LRRM.

3.1 RRMP Protocol

RRMP is a protocol based on the approach of distributed error recovery. The receivers are divided into a number of regions based on their distance from the sender. The regions are arranged in a tree-like hierarchy. Figure 1 shows one such hierarchy. A given receiver maintains information about two regions, the local region, of which it is a part, and the remote parent region, which is its immediate upstream region. For example, a member of Region 2 would maintain group information about the Region 2 as well as its parent region i.e. Region 1. Receivers maintain group membership information of their local as well as parent region by exchange of session messages.

Message loss is detected by discontinuity of the sequence numbers or by exchange of session messages. The error recovery algorithm in RRMP consists of two phases – local and remote. On detecting a message loss, a receiver concurrently initiates the local and the

remote recovery procedure. In local recovery, a receiver p randomly chooses another member q of its region and sends a request for transmission of the lost message. Simultaneously, a timer τ_{DAT} is started with a timeout value $TOut_{DAT}$. If q has buffered the message, it responds by sending a unicast reply. Otherwise, p's timer runs out and it chooses another member for a retransmission request. In remote recovery, p chooses a member r at random from its parent region and requests for the lost message. If r has the lost packet then it replies, else it notes down that p is waiting for a requested packet and initiates local recovery in its own region. When r receives the packet, it sends a unicast reply to p. When p receives a repair message from a remote member, it checks whether the message is a duplicate. If not, p multicasts the retrieved message in its local region. This is because remote recovery assumes that with a high probability no other member in the group has the message.

Receivers in RRMP have two types of buffers, *short term* and *long term*. All received messages are stored in the short term buffer for a certain time T. If no retransmission request is received for a message within a time interval T, a decision is made to either discard this *idle message* or store it in the long term buffer with a probability $P_{ShortToLong}$. The value of $P_{ShortToLong}$ is chosen in such a manner that the expected number of long term bufferers in a given local region is a constant C. Eventually, a message for which no retransmission requests have been received is deleted from the long term buffer. Implicitly, the short term buffer memory is considered a more precious resource, which offers faster recovery. Therefore, an increase in the hit ratio in the short term buffer would reduce the recovery latency. A more detailed description of the RRMP protocol and its performance benefits over tree based protocols can be found in [1].

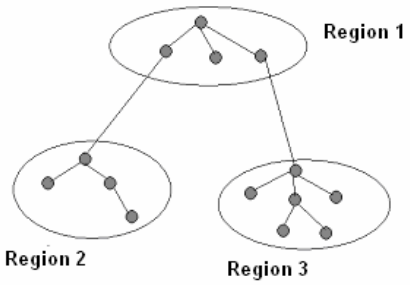


Figure 1. Hierarchy of regions used in RRMP and LRRM

Motivation for LRRM: RRMP makes extensive use of unicast requests to search for lost messages. To achieve reasonable recovery latency, a sizeable number of members C of a region must store messages in the long term buffer. The choice of C reflects a trade off between buffer requirements and recovery latency. Smaller values of C lead to longer recovery latencies. Further, if a large number of members lose messages, then a large number of requests for recovery will be generated. This will prevent the messages from being evicted from the buffers leading to high buffer usage.

3.2 Our Protocol: LRRM

We propose a protocol called the **L**ight-Weight **R**andomized **R**eliable **M**ulticast (LRRM) protocol that reduces the buffer requirements without impacting the recovery latency. LRRM is an extension of the RRMP protocol which builds on its various strengths. The novelty is that LRRM introduces an optimization in the message recovery algorithm used in RRMP. This is based on a parallel search for a local host for recovering lost messages unlike the serial search used by RRMP. This optimization can lead to lower buffer requirements at network nodes, which is useful for a wide class of resource constrained deployments of reliable multicast. However this optimization has to be harnessed carefully to prevent an explosion of replies and control messages and we show how we achieve this.

While the algorithm is defined in terms of RRMP, this technique can equally benefit other tree-based reliable multicast protocols that rely on a single or multiple hosts in a local region for fast local recovery. LRRM is also a tree based protocol with receivers grouped into regions. LRRM maintains information about local and parent region using session messages as in RRMP and has two recovery phases – local and remote.

Local Recovery: For local recovery in RRMP a node sequentially searches its neighbors to find a node that has the lost message in its buffers. Intuitively, it is known that querying all local neighbors simultaneously will lead to lower recovery latency. However, such an approach has the potential drawback of the overhead of multiple replies. We propose to use the multicast based approach for querying all the nodes in the local region in an intelligent fashion to counter this potential drawback leading to an overall performance improvement. Note that the multicast primitive being used is best-effort, while the primitive being provided is reliable multicast. For local recovery in LRRM, the node experiencing a message loss *multicasts* the recovery request to all the nodes in the local region. Multicasting helps avoid the delay associated with the sequential sending of unicast requests to the members of the region, as in RRMP. Any member of the local region that has the lost message sends a reply to the originator of the request. However, LRRM incurs the

additional cost of multicast tree formation whenever the membership of the local group changes.

Request Suppression: A problem which has daunted the use of multicast requests for error recovery is that it can lead to unwanted message flooding in certain cases. If two or more members simultaneously detect a message loss, they can make redundant multicast requests. To address this problem, LRRM uses a back-off algorithm for request suppression. When a node detects a message loss, it waits for a random time before multicasting a request for the message (Figure 2(a)). The choice of the back-off time depends on the following factors - local region size, region diameter, and the speed of the network links, which are combined into a maximum intra-region round-trip delay metric and a certain multiplicative factor of that is used in our protocol. Other members who hear a multicast request for a message that they themselves have not received, suppress their own multicast. Instead, each such node sends a unicast request to the original node which initiated the protocol for recovery of the message (Figure 2(b)). Now it becomes the responsibility of the originator to deliver this message to other requesting nodes once it retrieves the message (Figure 2(c)). If the originator node receives a large number of unicast requests exceeding a threshold δ_{REQ}, it multicasts the repair message in the local region. Otherwise, it sends unicast repair messages to each requesting node.

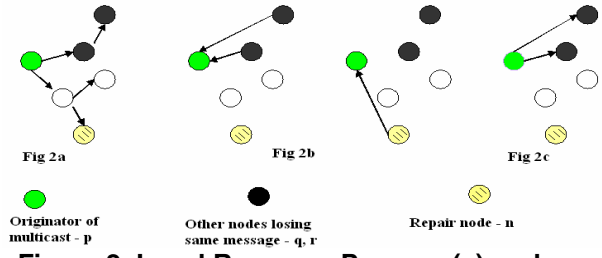


Figure 2. Local Recovery Process (a) node *p* loses the message and sends multicast request. (b) Nodes *q* and *r* that have lost the same message perform request suppression. (c) Node *p* gets the message from node *n* and sends unicasts to nodes *q* and *r*.

Another problem with using multicast requests for message recovery is duplicate replies which are received if more than one local node has the message in its buffer. This problem is handled in our message buffering scheme by probabilistically having at most one node in a region buffer a message. This is described later in this section.

Remote Recovery: Local recovery is able to satisfy a message loss request if even a single member has a copy of the requested message. However, in certain scenarios, such as localized congestion in a region of the network, all the members in a given region may miss a given message. To handle such scenarios, LRRM supports a remote recovery phase. In the remote recovery phase, the originator of the multicast request randomly chooses a member *M* from its parent region and sends a unicast request for the lost message. If *M* has the requested message, it services the request by a unicast reply; otherwise it follows the same multicast based local recovery algorithm in its region and services the request after it has acquired the message. This process can continue in a recursive fashion and in the worst case, the message can be retrieved from the original sender, which is guaranteed to have it.

To facilitate fast message recovery, the remote recovery phase is launched concurrently with the local recovery phase. The inter-region communication latency is typically several orders of magnitude larger than the intra-region latency. Further, sending too many remote recovery requests to the parent region may lead to unnecessary multicasts in the parent regions. Therefore, concurrent remote recovery phase is launched with a probability $P_{RemoteInitiate}$. The node waits for a timeout period $TOut_{Remote}$ after initiating the local recovery and if it does not get a response within the timeout, the remote recovery is deterministically initiated. The choice of parameter $P_{RemoteInitiate}$ depends on the average message loss rate, the size of the region, and the intra-region and inter-region round trip times. The parameter $P_{RemoteInitiate}$ can be adjusted according to application needs and environmental constraints. For example, in an environment with high loss rates and small buffer availability in the local region, $P_{RemoteInitiate}$ should be kept high to ensure reasonable recovery latencies.

Buffering Scheme: LRRM uses multicasting to service loss requests. This implies that, in a given local region, even a single copy of a lost message is capable of servicing the lost message request. This lowers the overall memory requirement and special long term buffers to augment buffer memory are not required. Therefore LRRM replaces the short and long term types of message buffers by a single type of buffer. To reduce the buffer requirements further, LRRM tries to maintain at most one copy of a message per local region. When a receiver receives a message, it uses a hash function on the message sequence number mapping it to a node ID to determine whether it should store the message. If the hash value matches the node's ID, it buffers the message with a probability P_{Buffer}. LRRM does not try to store copies of all the messages in a region because with moderate message loss rates, retransmission requests for most of the messages will never be issued. The value of probability P_{Buffer} is a

configurable parameter chosen based on message loss rates. LRRM assumes dynamic membership because of which the unique node which would have stored the message cannot be identified without storing the entire membership history. Thus unicast requests can't be used in place of multicast request to obtain a lost message as in [3].

4. Theoretical Analysis

In this section, we build a mathematical model for the recovery latency and the buffer utilization in RRMP and LRRM. We plot the two metrics using representative values of input parameters and compare the two schemes.

Table 1. Terminology used in LRRM and RRMP analysis

L	Average inter region latency	lo	Average intra-region latency
T_{Remote}	Time for remote recovery in a given region.	T_{Local}	Time for local recovery in a given region
P_{remote}	Probability of successful remote recovery	P_{local}	Probability of successful local recovery
V	Average region size	T_{lrrm_local}	Average time of local recovery in LRRM
T_f	Amortized time for multicast tree formation	T_m	Average time of a multicast request
T_r	Time to register a request for a lost packet with the original sender	T_b	Time for back-off
T_u	Time for unicast reply	D	Average depth of elements in multicast tree

4.1 Recovery Latency

In this analysis, we derive the recovery latency in LRRM and RRMP, taking both local and remote recovery into account. In a given local region, each node constructs a multicast tree with it being the source. To accommodate dynamic membership changes, a node reconstructs its multicast tree after every instance of node a joining or leaving in the group. We assume that such membership changes happen on an average once in every k message losses. The recovery latency in the local region, as shown in Section 3, is either (a) the sum of the times taken to back off, make a multicast request, receive a unicast reply from a repair node and once every k times, construct a local multicast tree or (b) the sum of the times taken to back off, register a request with the originator of a multicast request, receive a unicast reply from the originator of the multicast request, and once every k times, construct a local multicast tree.

Then the average local recovery latency is

$$T_{lrrm\_local} = 1/k\ [T_f + k\ \{T_b + (T_m \text{ or } T_r) + T_u\}]$$

In order to analyze the latency, we assume that the sender of any message in the protocol is at the root of a tree. For simplicity of analysis we consider the tree to be balanced. A balanced tree structure for the multicast tree leads to T_m being $O(\log V)$.The value of back off has been empirically chosen to be 4 times the maximum round trip time. Further, the cost of computing a minimum spanning tree knowing the costs of all graph edges using Prim's algorithm is $O(V^2 \log V + \log V)$. This assumes that the graph structure and edge costs are available in the form of an adjacency list representation of the graph. So, the computation of MST is to be preceded by all nodes reporting their adjacent lists. Since the actual packets containing the adjacency lists from all nodes will be small it can be assumed that these packets can be received simultaneously i.e. the time for uploading/downloading these packets is negligible compared to the latency in delivering these packets from individual nodes to the root of MST. Therefore, the latency will be given by the maximum latency among all the nodes. The worst case maximum latency for a graph with V vertices is O(V). So the total time for multicast tree formation is $O(V) + K_{proc}.O(V^2 \log V + \log V)$. The factor K_{proc} has been introduced to highlight the fact that the second term represents computation on a processor and time spent in this computation is a few orders of magnitude smaller than the first term that represents network delays. For most practical systems we may choose to ignore the second term without significant loss of accuracy. *Average value of* T_r and T_u for a balanced tree is O(logV) which is the average distance in number of hops from a node to the root, which is nothing but the average depth of the tree.

$$T_{lrrm\_local} = lo\ .\ 1/k\ .\ [V + (k\ (T_b + (\log V) + \log V)] = lo\ .\ 1/k\ .\ [V + k\ (4 + 2 \cdot \log V)]$$

The analysis in [1] states that the local recovery latency for RRMP is O(log V). This assumes that all the nodes which are unable to service a request join in the querying phase searching for the lost message. But this incurs the overhead that each node needs to maintain state about all the requests that it has been unable to satisfy and must continuously query for that lost message. Once the source gets the message from some node, it should send out a multicast and ask all other queries to quiesce. However, the protocol as described in [1] does not do this. A node which does not have a message being asked of it simply ignores the request. The average path length to neighbors is

O(logV) and on an average successful recovery occurs after V/2 queries.

$T_{rrmp\_local} = (V \cdot logV \cdot lo) / 2$.

Since, the recovery latency depends on both the local and remote recovery phases, the time of recovery at level *d*, given by T^d, is

$T^d = (P_{local} . T_{local}) + (P_{remote} . T_{remote})$ where $T_{remote} = T^{d-1} + L$

Recovery Latency for LRRM

Recovery latency in LRRM is given by

$T_{lrrm} = (P_{local} . T_{lrrm\_local}) + (P_{remote} . T_{lrrm\_remote}) = P_{local} . (lo .1/k .[V+ k (4+2 \cdot logV)]) + P_{remote} . (T^{d-1} + L)$

As P_{remote} is a small value and we assume a uniform local region size of V, we can ignore terms of this recurrence relation beyond one level higher. Then,

$T_{lrrm} = P_{local} (lo .1/k .[V+ k (4+2 \cdot logV)]) + P_{remote} \cdot (P_{local} . (lo.1/k .[V+ k (4+2 \cdot logV)]) + L)$ — **(1)**

Recovery Latency for RRMP

For RRMP, $T_{rrmp\_local} = (VlogV \cdot lo)/2$ and $T_{rrmp\_remote} = L + (logV . lo)$. The time for remote recovery is log V instead of V·logV because when a node in a parent region gets a request to recover a message, it uses a protocol based on epidemic theory to spread out the query. Again ignoring the possibility of having to query more than one level higher up, we have

$T_{rrmp} = P_{local} . (V.logV.lo) / 2 + (P_{remote} . [L + (logV.lo)])$ — **(2)**

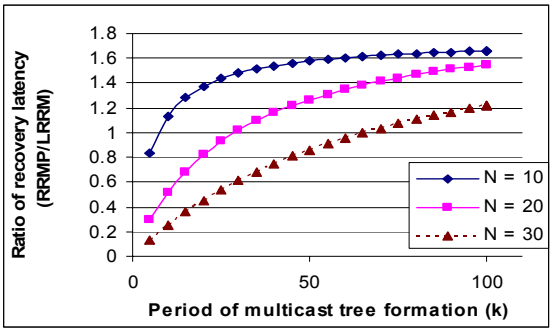


Figure 3. Ratio of recovery latencies against period of multicast tree formation

Using the two expressions (1) and (2), we plot the ratio of recovery latencies for RRMP and LRRM with the frequency of changes in network topology as the control parameter. The values of the input parameters are *lo* = 5ms and *L* = 50ms. The number of long term bufferers for RRMP is chosen such that its buffer utilization is the same as that of LRRM.

Figure 3 shows that for a given region size as the frequency of creation of the multicast tree is reduced, i.e., the environment becomes less dynamic with respect to node joins and leaves, LRRM performs better. The cost of multicast tree formation increases with the increase in the region size and consequently the results get biased against LRRM. However, for reasonable group sizes (~20), the crossover point is still low enough (~30 lost messages), that LRRM can handle fairly dynamic environments. It must be noted that incorporating the cost of constructing a multicast tree is a fairly pessimistic estimate of the performance of LRRM. Typically, the local regions in LRRM will map to administrative domains each on a separate LAN. In such a scenario, LRRM does not have to construct a multicast tree. It can use broadcast instead, since broadcast does not consume any more network resource than unicast or multicast on a typical Ethernet based LAN.

4.2 Buffer Utilization

Here we derive the buffer capacity needed to guarantee a given recovery latency. Let the node which has lost a message be *X*. Further, let the number of long term bufferers in the region be *C*. The probability that *X* gets the message in the first try is $P_1 = C/V$. The probability that *X* gets the message in the second try is $P_2 = (1-C/V) . C/(V- 1)$.

Proceeding in this fashion, the probability that *X* will get the message in the k^{th} try is

$$P_k = (\prod_{i=0}^{k-2} 1 - C/(V-i)) . C/(V-k+1)$$

In the worst case, the node contacts the repair node in the $(V-C)^{th}$ attempt, assuming it contacts the *(V-C-1)* nodes (except itself) that do not have the message before getting lucky. So, the value of *k* in the above series goes till *(V-C)*.

The expected number of attempts thus will be

$$E[tries] = \sum_{i=1}^{V-C} (i * P_i)$$

$T_{rrmp\_local} = (E[tries] - 1). [lo + TOut_{DAT}] + (2.lo)$

$T_{lrrm\_local} = Back\ off + (Multicast\ request\ or\ registering\ a\ request) + unicast\ reply$

Using the values of the parameters, as specified before, we plot the local recovery latencies of both the protocols in Figure 4. RRMP needs more than 30% of the group members to be long term bufferers to perform better than LRRM for local recovery. LRRM's recovery latency is sensitive to the back off time chosen for the protocol but not on the number of bufferers in a region. In fact, the design of LRRM enforces that at most one node in a region will buffer a packet. The worst case back-off period is chosen conservatively to be 4 times the intra-region latency for this analysis. Also the analysis for LRRM does not model request suppression to simplify the analysis. Recovery at nodes which suppress their repair request is only delayed by time it takes the designated receiver to send a unicast message i.e. *lo*.

The above analysis only took local recovery into account. Next, we analyze the total buffer requirements of both the protocols to achieve a given recovery latency, taking both local and remote recovery into account. We assume a relatively static environment so that the multicast tree formation is a one-time cost incurred in LRRM. We use a region size of 30. The recovery latency for both the protocols will depend upon the success of local recovery phase.

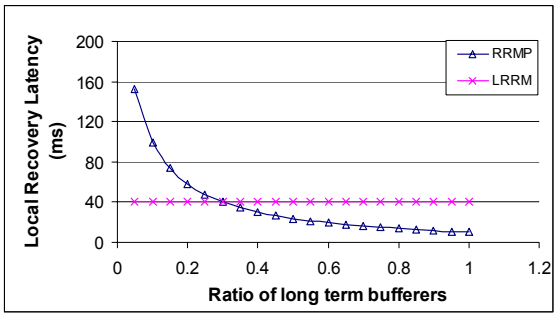
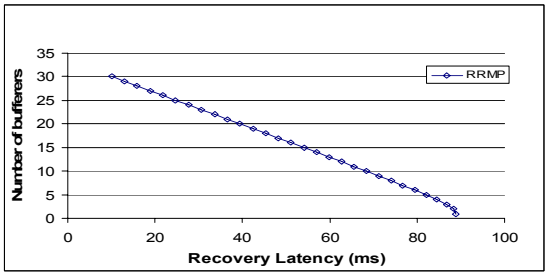
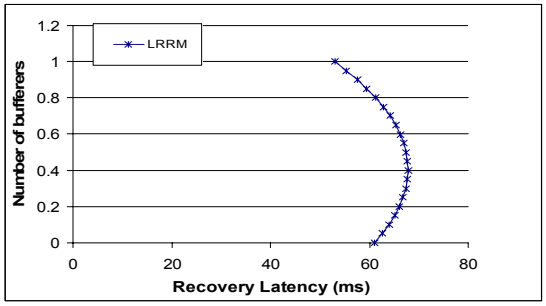


Figure 4. Local recovery latencies with ratio of long term bufferers

For RRMP, we use the probability of success of the local recovery phase $(P_{local} = C/V)$ as an input parameter to obtain a set of recovery latencies. For LRRM, we use the probability of buffering a message (P_{Buffer}) as the input parameter to obtain a different set of recovery latencies. The y-axis gives the number of bufferers; for RRMP, this is $C\ (1 \le C \le V)$, for LRRM this is $P_{Buffer}\ (0 \le P_{Buffer} \le 1)$. As shown in Figure 5, RRMP needs 33% of its members to act as long term bufferers to outperform LRRM. An interesting observation for LRRM is that the recovery latency expectedly increases as P_{Buffer} is reduced, but below a probability of 0.4, it decreases. In these experiments, the inter-region latency is one order of magnitude higher than the intra-region latency. Therefore, for very frequent local recoveries (high P_{Buffer}), the multicast tree formation cost causes the overall performance to degrade. If the inter-region latency was taken as several orders of magnitude higher as is common in most WANs, then this phenomenon will not be observed.



(a)



(b)

Figure 5. Buffer Requirements Vs Recovery Latency with group size=30 (a) RRMP (b) LRRM

The non monotonicity in the plot can also be explained mathematically.

$T_{LRRM} = P_{Buffer} \cdot T_{local} + (1\text{-}P_{Buffer}) \cdot [L + P_{Buffer}.T_{local} + \ldots higher\ terms\ ignored]$

$= P_B \cdot T_l + (1\text{-}P_B) \cdot (L + P_B.T_l) = L + (2T_l - L) \cdot P_B - T_l \cdot P_B^2$

Differentiating this expression w.r.t P_B, we get an optima at $P_B = 1 - L/2T_l$ This is a maxima since the double differential is negative. For this analysis, L = 50 ms, T_{local} = 45-70 ms. Hence, the maxima is reached for a $P_B > 0$. With a larger value of L, the maxima will be unreachable in practice.

5. Simulation

In this section, we provide detailed comparative simulation studies of LRRM and RRMP with models built in ns-2 [6].

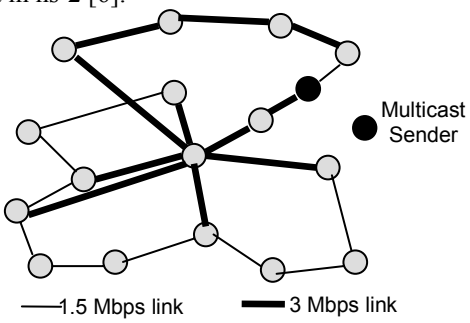


Figure 6. Topology A represents a real-life network that connects research institutions in South Eastern USA [7]

Two different network topologies are modeled. The first topology, henceforth called Topology A, as shown in Figure 6 represents a real-world topology of 17 nodes closely based on the SURAnet backbone, a contemporary WAN interconnecting research institutions in the Southeastern US [7].The second topology, henceforth called Topology B, is the transit-

stub network topology[1] of 50 nodes, generated using the Georgia Tech Internet Topology generator (GT-ITM) [8]. Links within the transit domains have a bandwidth of 3 Mbps. Links connecting the transit domains to stub domains and links within the stub domains have a bandwidth of 1.5 Mbps. The inter region delay (L) is set to 50 ms, while the intra-region latency (l) is set to 5 ms.

In the SURAnet topology, traffic in the individual routing domains of the research institutions is not modeled. The simulation only considers a multicast scenario where a multicast stream from one institution is being broadcast to a single host in each other university. This represents a real world situation where a major event at an institution like a lecture by a distinguished personality is being webcast to other institutions on the SURAnet. Topology B represents a webcast scenario over a WAN or over the internet. The network consists of multiple LANs and several hosts within a LAN (but not all) are receiving the webcast. This models a situation where a university event is being webcast to student computers in dorms and laboratories.

To make our simulations realistic we also assume background traffic in the networks. Constant Bit Rate (CBR) traffic and ftp traffic both within LANs and between different LANs is modeled. A message loss probability of 5% is assumed for all multicast messages. The simulation is conducted with one randomly chosen node as the originator of multicast traffic that is reliably delivered to all other nodes in the network.

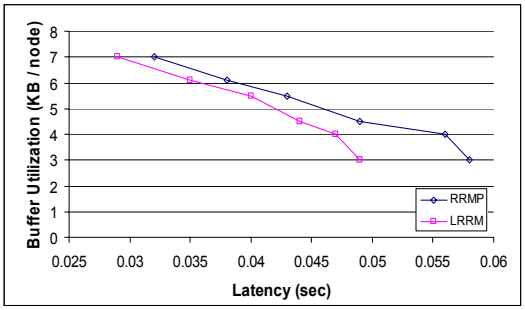
Table 2. Simulation parameters for LRRM

Intra (l) and Inter (L) region delays	5 ms, 50 ms
Packet loss rate (r)	0.05
Back-off period	4.lo
Time period for eviction	4.RTT
Packet size	500 bytes

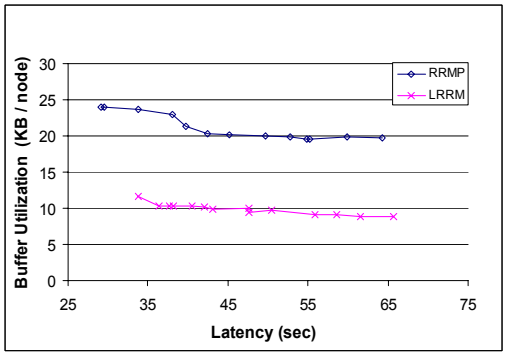
Table 3. Simulation parameters for RRMP

Intra (l) and Inter (L) region delays	5 ms, 50 ms
Packet loss rate (r)	0.05
Probability of being a long term bufferer	25%
Packet size	500 bytes

For the remainder of this section, the results from the two topologies are presented together. Sub-figure (a) represents the results for topology A and (b) for topology B. As we saw from the theoretical analysis, LRRM has lower buffer requirements compared to RRMP to achieve a given recovery latency. However, the analysis made simplifying assumptions – it used asymptotic complexities and costs (big-O notation), considered only long term buffering in RRMP, and ignored the possibility of a message request being satisfied by a predecessor region higher than the immediate parent. To validate our claims without these assumptions, we run the simulation model to estimate the buffer requirement for a given recovery latency (simulation parameters in Table 2 and Table 3). The buffer requirement is given by the average number of bytes in the buffer per node.



(a)



(b)

Figure 7. Buffer utilization required to achieve a given recovery latency

Figure 7(a) shows that the buffer requirement in topology A increases rapidly with lower recovery latency requirements. The recovery latency for LRRM cannot go below 0.03 sec and that for RRMP below 0.035 sec. For a given recovery latency in the range that is common to the two protocols, the buffer requirement in RRMP is between 10-20% higher than in LRRM. For topology B, the buffer utilization does not increase so sharply and a large common set of recovery latencies is achievable. The difference in the buffer requirements for RRMP and LRRM is substantially higher, with LRRM needing on average 50% less buffer space for a comparable latency. The

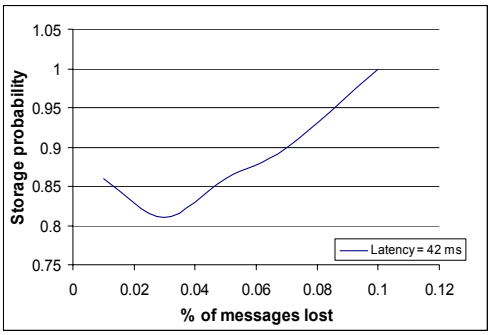
[1] Stub domain in an internet carries only traffic that originates or terminates in that domain. Transit domains do not have this restriction. Transit-stub topologies are two level hierarchical graphs generated by interconnecting transit and stub domains.

reason for the improvement of LRRM for topology B is that the topology has an inherent tree structure, which leads to an efficient distribution of nodes into local and remote regions. Since LRRM relies on minimizing the number of copies of a message in a local region, a more efficient local region demarcation helps in optimal buffer utilization.

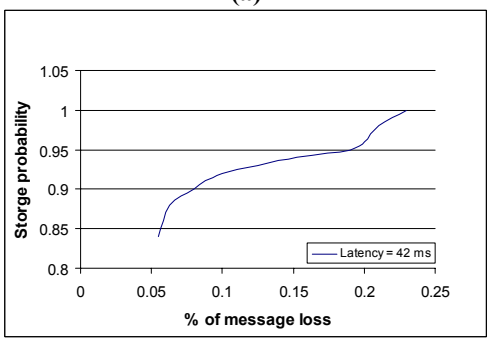
The reduction in buffer utilization by LRRM has an associated cost in increased control message overhead. All messages other than the multicast data messages, such as recovery request and recovery reply messages, are considered control messages. For LRRM, the control overhead includes the recovery request transmitted along the edges of the multicast tree with a message on each hop being counted as one overhead message. However, it does not include the control messages for *forming* the multicast tree. This would be a valid simplification only for fairly static environments. For topology A, the control message overhead for RRMP is 36.8% (625 control messages for 100 multicast messages each to 17 receivers) while for LRRM it is 42.3%.

Trade off between packet loss ratio and storage probability: In resource constrained networks with low error rates, it may be desirable not to buffer all the messages in a given region. LRRM has a configurable parameter P_{Buffer} which gives the probability of storing a message at a node if the hash value of the message matches. Reducing P_{Buffer} can further reduce the buffer requirement for the operation of the protocol. In this experiment, we show the relationship between the message loss rate and the value of P_{Buffer} needed to maintain the recovery latency at a pre-specified level, 42 ms here. This experiment is carried out for both topologies. The result for topology B has no surprises and the storage probability increases quite uniformly with loss rate. It is seen that to maintain a latency of 42 ms, the packet loss rate has to be less than 22%. For packet losses between 22% and 5% the probability of storage reduces to a minima of 0.84.

The result for topology A is interesting. Figure 8(a) shows that to achieve the required latency of 42 ms, the packet loss rate must be less than 10%. For packet loss rates between 3% and 10%, the probability of storage reduces linearly to a minima of 0.81. An interesting phenomenon is observed for packet loss rate below 3%. In such a scenario, packet failures are so rare that all buffered packets become idle and are removed. So if a member loses a packet then the recovery almost always occurs through the costly remote recovery procedure. Therefore, to maintain comparable recovery latency, we need to increase the storage probability. This is in line with the earlier observation about the lack of an inherent tree structure in this topology leading to less efficient clustering of nodes into local regions.



(a)



(b)

Figure 8. Variation of storage probability (P_{Buffer}) of message in LRRM with packet loss rate to maintain given recovery latency

6. Conclusion and future work

Multicasting is an important technology for disseminating data to multiple receivers. Adding reliability to multicasting requires buffering of messages at participating nodes. Making efficient use of constrained resources like message buffers is a challenging problem. We have presented a protocol called LRRM, which is a lightweight version of the RRMP protocol for reliable multicasting. LRRM addresses the challenge of optimizing the buffer requirement at repair nodes without impacting the recovery latency. LRRM achieves this by using best-effort multicast for disseminating the request for a lost packet to nodes within its region and suppressing redundant requests and replies. RRMP has already been demonstrated to have better recovery latency and reliability compared to other tree based protocols such as TRMP and this benefit is retained for LRRM. The disadvantage of LRRM compared to RRMP is that it is unsuitable in very dynamic environments with frequent node joins and leaves. Both analytical and simulation results are presented, with simulation being performed for two real-world topologies. The results for topology A show buffer requirement reduction of 10-25% and

that for topology B, 100-125%. The simulation results also show that the buffer requirement to maintain a given recovery latency scales down gracefully in LRRM as the message loss rate in the environment decreases.

The current focus of work is on modifying LRRM to use some of the features of epidemic protocols, and optimizing when and where remote recovery is initiated based on prediction of availability of a message in the local region.

7. References

[1] Zhen Xiao and K. Birman, "A Randomized Error Recovery Algorithm for Reliable Multicast", IEEE Infocom, April 2001, pp. 239-248.

[2] Zhen Xiao, K.P. Birman, R. van Renesse, "Optimizing buffer management for reliable multicast", IEEE International Conference on Dependable Systems and Networks (DSN '02), June 2002, pp. 187-198.

[3] Oznur Ozkasap, Robbert van Renesse, Kenneth Birman, and Zhen Xiao, "Efficient Buffering in Reliable Multicast Protocols", Proceedings of the First Workshop on Networked Group Communication. (NGC99) Pisa, Italy. (November 1999), pp. 188-203.

[4] Sanjoy Paul and John C. Lin, "RMTP: A Reliable Multicast Transport Protocol", IEEE Infocom 1996, pp.1414-1424.

[5] Dah Ming Chiu, Stephen Hurst, Miriam Kadansky and Joseph Wesley, "TRAM: A Tree-based Reliable Multicast Protocol", Sun Technical Report TR 98-66, July 1998.

[6] "The network simulator ns-2," At: http://www.isi.edu/nsnam/ns/

[7] M. T. Lucas, B. J. Dempsey, and A. C. Weaver. "MESH: Distributed Error Recovery for Multimedia Streams in Wide-Area Multicast Networks," In Proceedings of IEEE International Conference on Communication (ICC '97), pp. 1127-1132, June 1997.

[8] GT-ITM: Georgia Tech Internet Topology Models. www.cc.gatech.edu/fac/Ellen.Zegura/graphs.html

[9] Hugh W. Holbrook, Sandeep K. Singhal, David R. Cheriton, "Log-Based Receiver-Reliable Multicast for Distributed Interactive Simulation", SIGCOMM 1995, pp. 328-341.

[10] Jun Lo, Patrick Th Eugster, Jean Pierre Hubaux, "Route Driven Gossip: Probabilistic Reliable Multicast in Ad Hoc Networks", IEEE Infocom 2003, pp. 2229 –2239.

[11] Rajendra Yavatkar, James Griffioen, and Madhu Sudan, "A Reliable Dissemination Protocol for Interactive Collaborative Applications", ACM Multimedia, 1995, pp. 333-344.

[12] Kenneth P Birman, Mark Hayden, Oznur Ozkasp, Zhen Xiao, MIhai Budiu and Yaron Minsly, "Bimodal Multicast", ACM Transactions on Computer Systems, May 1999, vol. 17, no. 2, pp. 41-88.

[13] Puneet Sharma, Deborah Estrin, Sally Floyd, and Lixia Zhang, "Scalable session messages in SRM using self-configuration," Tech. Report, University of Southern California, 1998.

[14] S. Floyd, V. Jacobson, C. Liu, S. McCanne, and L. Zhang, "A Reliable Multicast Framework for Light-weight Sessions and Application Level Framing", IEEE/ACM Transactions on Networking, December 1997, Volume 5, Number 6, pp. 784-803.

[15] R. Chandra, V. Ramasubramanian, K. Birman, "Anonymous Gossip: Improving Multicast Reliability in Mobile Ad-Hoc Networks," In Proceedings of the 21st Intl Conference on Distributed Computing Systems, ICDCS 21, 2001, pp.275-283.