Optimizing Buffer Management for Reliable Multicast*

Zhen Xiao
AT&T Labs – Research
180 Park Avenue
Florham Park, N.J. 07932
xiao@research.att.com

Kenneth P. Birman, Robbert van Renesse
Department of Computer Science
Cornell University
Ithaca, N.Y. 14853
{ken, rvr}@cs.cornell.edu

Abstract

Reliable multicast delivery requires that a multicast message be received by all members in a group. Hence certain or all members need to buffer messages for possible retransmissions. Designing an efficient buffer management algorithm is challenging in large multicast groups where no member has complete group membership information and the delivery latency to different members could differ by orders of magnitude.

We propose an innovative two-phase buffering algorithm, which explicitly addresses variations in delivery latency seen in large multicast groups. The algorithm effectively reduces buffer requirements by adaptively allocating buffer space to messages most needed in the system and by spreading the load of buffering among all members in the group. Simulation and experimental results demonstrate that the algorithm has good performance.

Keywords*: reliable multicast, buffer management, error recovery, randomization, scalability*

1 Introduction

Multicast is an efficient way for disseminating data to a large group. Many emerging multicast applications require reliability guarantees not provided by the IP multicast protocol [4]. Providing reliable multicast service on a large scale requires an efficient error recovery algorithm. It has been shown that putting the responsibility of error recovery entirely on the sender can lead to a message implosion problem [7, 13]. Consequently, several reliable multicast protocols adopt a distributed error recovery approach which allows certain or all members to retransmit packets lost by other members. For example, in the SRM protocol [7], the Bimodal Multicast protocol [2], and the Randomized Reliable Multicast Protocol [16], retransmissions are performed by all members in the group. In tree-based protocols like RMTP [13], LBRRM [9], and TMTP [17], members are grouped into local regions based on geographic proximity and a repair server is selected in each region to perform retransmissions.

If a member wants to perform retransmissions for other members, it needs to buffer received messages for some period of time. Determining which receivers should buffer a message and for how long turns out to be a difficult problem. A conservative approach is to have every member buffer a message until it has been received by all current members in the group. However, this is inefficient in a heterogeneous network where the delivery latency to different members could differ by orders of magnitude. Moreover, some reliable multicast protocols adopt the IP multicast group delivery model in which receivers can join or leave a multicast session without notifying other receivers. Consequently, no single receiver has complete membership information about the group.

Buffer management algorithms in existing reliable multicast protocols reflect widely different strategies for deciding which members should buffer messages and how long a message should be buffered. In some tree-based protocols, a repair server buffers all data packets it has received in the current multicast session. For example, the RMTP protocol was originally designed for multicast file transfer. In this protocol, a repair server buffers the entire file in a secondary storage. This approach is feasible only if the size of data transmitted in the current session has a reasonable limit. For long-lived sessions or settings where repair servers lack space, the amount of buffering could become impractically large.

The SRM protocol does not buffer packets at the transport level. Rather, the application regenerates packets if necessary based on the concept of Application

*This work was supported in part by DARPA/RADC under grant number F30602-99-1-0532. Any opinions, findings, or recommendations presented in this paper are those of the authors and do not reflect the official views of the funding agencies.

Level Framing (ALF) [3]. This requires that the application be designed according to the ALF principle and is capable of reconstructing packets. Even so, buffer management at the application level remains a challenge.

Some reliable multicast protocols use a stability detection algorithm to detect when a message has been received by all members in the group and hence can be safely discarded [8]. This requires members in the group to exchange message history information periodically about the set of messages they have received. In addition, a failure detection algorithm is needed to provide current group membership information.

Previously we proposed a message buffering algorithm for reliable multicast protocols that reduces the amount of buffer requirement by buffering messages on only a small set of members [12]. More specifically, we assume that each member has an approximation of the entire membership in the form of network addresses. The approximation needs not be accurate, but it should be of good enough quality that the probability of the group being logically *partitioned* into disconnected subgroups is negligible. Upon receiving a message, a member determines whether it should buffer the message using a hash function based on its network address and the identifier of the message. (A commonly used identifier is [source address, sequence number].) If a member missed a message, it uses the same hash function to identify the set of members which should have buffered the message and requests a retransmission from one of them.

This algorithm makes no use of network topology information. Consequently, it suffers from a tendency to do error recovery over potentially high latency links in the network. The probability of this happening and the associated penalty in latency both increase with the size of the group. Hence the protocol will have a scalability problem in very large networks. It is desirable to have an algorithm that selects receivers to buffer a message based on geographic locations of different receivers. Unfortunately, our previous algorithm cannot be easily modified to incorporate such information. The work described here was motivated by this observation.

In this paper, we report our work on optimizing buffer requirements for a randomized reliable multicast protocol called RRMP. The error recovery algorithm in RRMP combines our previous work on randomized error recovery in Bimodal Multicast [2] and hierarchical error recovery similar to that employed by tree-based protocols. Its buffer management extends our previous work on buffer optimization by proposing an innovative two-phase buffering algorithm that explicitly addresses the variances in delivery latency for large multicast groups. The algorithm reduces buffer requirements by adaptively allocating buffer space to messages most needed in the system and by spreading the load of buffering among all members in the group. Unlike stability detection protocols, the algorithm does not require periodic exchange of messages and has low traffic overhead.

The rest of the paper is organized as follows. In Section 2 we briefly describe the error recovery algorithm in RRMP because it is closely related to the work reported here. A complete description can be found in [16]. Readers already familiar with the protocol can proceed directly to Section 3, where we describe the details of our buffer management algorithm. Sections 4 and 5 evaluate its performance using simulation and experiments. Limitations of the algorithm are presented in Section 6. Section 8 concludes.

2 A Randomized Error Recovery Algorithm

The RRMP protocol is designed for multicast applications with only one sender. We assume that receivers are grouped into local regions based on their geographic locations and that different regions are organized into a hierarchy according to their distance from the sender. This is called the *error recovery hierarchy*. Figure 1 shows an example of a hierarchy where the whole group is divided into three local regions. We define the *parent* region of a receiver as its least upstream region in the hierarchy. For example, in Figure 1, region 1 is the parent region for all receivers in region 2. If a receiver is in the same region as the sender, then it has no parent region.

Each receiver maintains group membership knowledge about other receivers in its region as well as receivers in its parent region. This is achieved by periodic exchanges of session messages among all members in a group. We adopt an idea from the scalable session message protocol [14] in SRM which divides session messages into two categories: *local* session messages and *global* session messages. Local session messages are multicasts restricted within a local region and global session messages are multicasts that reach the entire group. Session messages are also used to synchronize state among receivers and to help a receiver detect the loss of the last message in a burst, an idea previously used in the SRM protocol. In [16] we describe an algorithm to construct the error recovery hierarchy and to provide the required group membership knowledge at each receiver.

In RRMP, the responsibility of error recovery is distributed among all members in the group. Its error recovery algorithm consists of two phases executed concurrently: a local recovery phase and a remote recovery phase. In the local recovery phase, when a receiver p de-

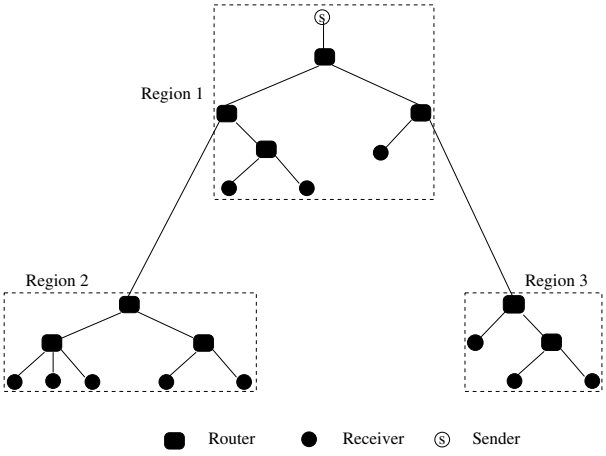

Figure 1. Local regions in a hierarchical structure

tects a message loss, it selects a receiver q uniformly at random from all receivers in its local region and sends a request to q. p also sets a timer according to its estimated round trip time to q. (Round trip time measurements are described in [16].) Upon receiving p's request, q checks whether it has the message. If so, it sends the message to p. Otherwise it ignores the request. If p does not receive a copy of the message when its timer expires, it randomly selects another receiver in its region and repeats the above process. As long as at least one local receiver has the message, p is eventually able to recover the lost message.

In the remote recovery phase, when a receiver p detects a message loss, it randomly chooses a remote receiver r in its parent region and, with a small probability, sends a request to r. This probability is chosen so that the expected number of remote requests sent by all receivers in the region is a constant λ. p also sets a timer according to its estimated round trip time to r. This timer is set by any receiver missing a message, regardless whether it actually sent out a request or not. If p does not receive a copy of the message when its timer expires, it randomly selects another receiver in its parent region and repeats the above process.

Upon receiving a request from a remote receiver p, r checks whether it has the requested message. If so, it sends the message to p. Otherwise, r missed the message as well. In this case, r records "member p is waiting for the message". When r later receives a copy of the message, it relays the message 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 message in its local region so that other members sharing the loss can receive the message.

The two phases described above, local recovery and remote recovery, are executed concurrently when a receiver detects a message loss. If a receiver has no parent region, its remote recovery phase does nothing.

3 Optimizing Buffer Management

As described in the previous section, the RRMP protocol distributes the responsibility of error recovery among all members in a group. Hence every member needs to decide how long a message should be buffered for possible retransmissions. The problem is that this involves a trade-off with error recovery latency. If a member discards a message and later receives a retransmission request for that message, it will be unable to answer the request. Due to the randomized nature of our error recovery algorithm, this does not necessarily compromise the correctness of the protocol because another request will be sent to a randomly chosen member upon timeout. As long as some member still buffers the message, the loss can be recovered eventually. However, error recovery latency is increased because more requests were needed to repair the loss. The problem is even more complicated in a wide area network where the latency between two regions can be significantly higher than the latency within a region. Since a member can receive a request either from a local member or from a remote member, it is difficult to determine how long a message should be buffered for potential requests.

In order to reduce buffer requirements effectively while minimizing its impact on recovery latency, RRMP adopts an innovative two-phase buffering scheme: feedback-based short-term buffering and randomized long-term buffering. When a message is first introduced into the system, every member that receives the message buffers it for a short period of time in order to satisfy local retransmission requests. Later when the message has been received by almost all members in a region, only a small subset of members in this region continue to buffer the message. The rest of the section describes the details of our scheme.

3.1 Feedback-based Short-term Buffering

First we investigate how long a member should buffer a message for local retransmission requests. Since the outcome of the initial IP multicast for each message can be different, it is undesirable to buffer every message for the same amount of time. For example, if only a small fraction of members in a region have received a message during the initial IP multicast, these members should buffer the message for a long period of time in

order to satisfy local requests from other members. In contrast, if almost all members have received the message during the initial multicast, then the message can be discarded quickly. Ideally, we want to allocate buffer space to messages most needed in the system.

In RRMP, the buffering time for a message is based on an estimation of how many members in the region have received the message. One way to estimate this information is to let all members periodically exchange message history information about the set of messages they have received, an idea previously used in some stability detection protocols [8]. Here we propose a different scheme in which a member estimates this information based on the history of retransmission requests it has received. Recall that in RRMP every member missing a message sends local requests to randomly selected members in its region. Hence the likelihood that a member receives a request increases with the number of members missing the message. More formally, let n be the size of a region and p be the percentage of members in this region missing a message. The probability that a member will *not* receive any request is:

$$(1 - \frac{1}{n-1})^{np}$$

As $n \to \infty$, this probability can be approximated by e^{-p}, which decreases exponentially with p. Consequently, if a member has not received any request after a sufficiently long period of time, it can conclude with high confidence that almost all members in the region have received the message. Based on this observation, we design a *feedback-based* scheme for short-term buffering: when a member receives a message, it buffers the message until no request for this message has been received for a time interval T. Such a message is called an *idle* message and T is called the *idle threshold*. The choice of T depends on the maximum round trip time within a region and the confidence interval. We call this a *feedback-based* scheme because a member uses the retransmission requests it received as feedback to estimate how many members in the region still miss the message. Unlike stability detection protocols, our scheme does not introduce extra traffic into the system.

3.2 Randomized Long-term Buffering

After a message has become idle, a member may decide to discard it. However, due to the randomized nature of the algorithm, it is possible that a message is still missing at some receivers but has become idle everywhere else. These unlucky receivers will not be able to recover the loss if all other members have decided to discard the message. Moreover, since inter-region latency can be much larger than intra-region latency, a member may receive a remote request from a downstream member asking for a message which has become idle at all members in the region.

RRMP addresses this problem by providing long-term buffering for an idle message at a small subset of receivers in each region. The set of long-term bufferers are chosen randomly from all members in a region. More specifically, when a member detects that a message has become idle, it makes a random choice to become a long-term bufferer with probability P. P is chosen so that the expected number of long-term bufferers in the region is a constant C. For a region with n members, probability theory shows that the number of long-term bufferers has a binomial distribution with parameters n and P [5]. As $n \to \infty$, $P \to 0$ and $nP \to C$. Hence for large regions the distribution can be approximated by a Poisson distribution with parameter C. (In [16] we applied a similar technique to analyze the number of remote requests sent when an entire region missed a message.) The probability that k members buffer an idle message is $e^{-C}\frac{C^k}{k!}$. Figure 2 shows how the distribution changes with different values of C. The choice of C reflects a trade-off between buffer requirements and recovery latency. With large C more members buffer an idle message, and hence an unlucky receiver in the previous scenario will recover the loss faster. On the other hand, small C reduces buffer requirements but may lead to longer recovery latency. In particular, it is possible that an idle message is buffered nowhere due to randomization. The probability of this happening decreases exponentially with C as shown in Figure 3. When $C = 6$, for example, the probability is only 0.25%. (This is the probability that no receiver in a region buffers a message in its long-term buffer. It is *not* the probability that a receiver will miss a message. For example, if the receiver gets the message during the initial multicast, it will not need any error recovery at all.)

When the sender multicasts a stream of messages, the load of long-term buffering is spread evenly among all members in a region. This is in contrast to some tree-based protocols where a repair server bears the entire burden of buffering messages for a local region. Eventually even a long-term bufferer may decide to discard an idle message if the message has not been used for such a long time that it is highly unlikely any member may still need it.

Receivers may join or leave a multicast session dynamically. When a receiver voluntarily leaves the group, it transfers each message in its long-term buffer to a randomly selected receiver in the region. This avoids the situation where all long-term bufferers decide to leave

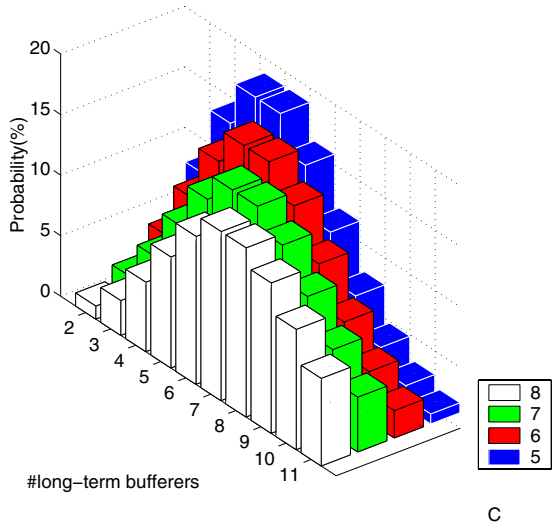

Figure 2. For large regions, the number of long-term bufferers for an idle message approximately follows a Poisson distribution with parameter C.

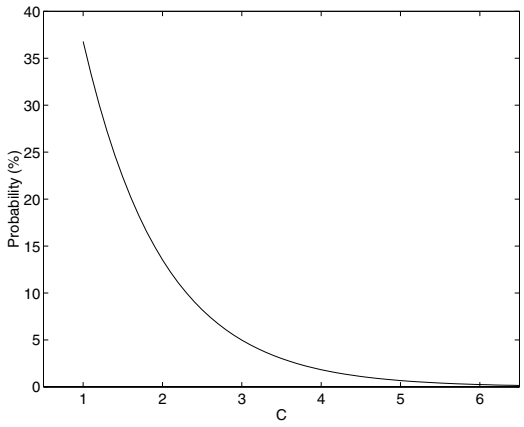

Figure 3. For large regions, the probability that no member buffers an idle message decreases exponentially with C.

the group, making a message loss unrecoverable.

3.3 Search for Bufferers

When a member p receives a remote request from a downstream member r for a message, there are three possibilities:

- p has received the message and still buffers it.
- p has never received the message.
- p received the message but has discarded it.

In the first case, p can immediately send the message to r. In the second case, p records r's request. Later when p receives the message, it will forward the message to r as described in Section 2. In the third case, however, p needs to search for a member which buffers the message.

One solution is for p to multicast r's request in its region. If a member has the message in its buffer, it multicasts a reply "I have the message" and then forwards the message to r. A randomized back-off scheme is used to suppress duplicate responses when multiple members buffer the message: upon receiving a request, a member waits a random amount of time before multicasting its reply in the region. If it hears a multicast for the same message from another member, it suppresses its own multicast. The problem is how to choose an appropriate back-off period. As described earlier, the expected number of long-term bufferers for an idle message is C. Hence it is tempting to set the back-off period to be proportional to C. In practice, however, we have found that this approach occasionally leads to message implosion. Recall that in our feedback-based buffering scheme each member independently decides when a message has become idle based on retransmission requests it received from other members. Because of randomization, it is possible that a message has become idle and been discarded at one member but is still being buffered at many other members (i.e. the message has *not* become idle at all members in the region). If a multicast request is sent in this case, the back-off period will be too short to suppress duplicate responses effectively.

In order to avoid storms of multicast replies, RRMP adopts a different approach where a member conducts a random search in its region to find out a bufferer of the message. More specifically, when p receives r's request, it randomly selects a member q in its region and forwards r's request to q. p also sets a timer according to its estimated round trip time to q. Upon receiving r's request, q checks whether the message is still in its buffer. If so, it sends the message to r and multicasts a reply "I have the message" in its region. This reply notifies other members that the search process is over. If q has discarded the message as well, it joins p in the search process and tries to find a bufferer of the message. (If q has never received the message, it will send retransmission requests as described in Section 2.) If p does not hear a reply when its timer expires, it randomly selects another receiver in its region and repeats the above process. As time goes by, more and more members will join the search process. As long as at least one member in the region still buffers the message, r will receive the message eventually.

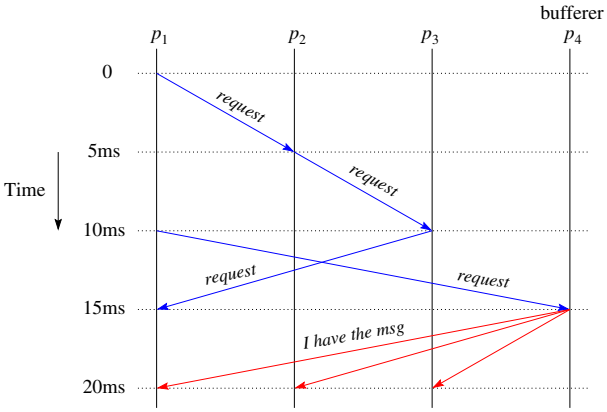

Figure 4. Search for bufferers in a local region

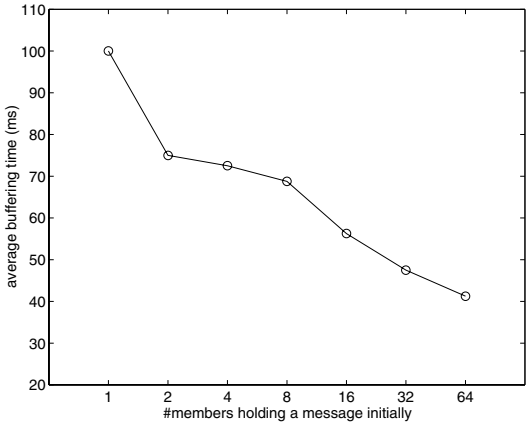

Figure 5. Effectiveness of feedback-based buffering. The x-axis is in logarithmic scale. The figure indicates that the amount of buffering time decreases as the initial IP multicast has reached more members.

Figure 4 illustrates the search process in a region with four members, one of which is a bufferer. The horizontal direction represents different members in the group, and the vertical direction represents the amount of time that has elapsed since the search starts. We assume that the latency between any two members in the region is 5ms. Suppose member p_1 receives a remote request at time 0. It forwards the request to a randomly selected member p_2. Since p_2 does not have the message either, it forwards the request to p_3. After 10ms p_1 times out and sends another request to p_4, which is the bufferer. Upon receipt of the request, p_4 sends the message to the remote member and multicasts a reply in the region.

The search time for a message depends on the number of members that buffer the message. If the message has become idle at all members in the region, the expected number of bufferers is C. Hence increasing C can reduce search time at the expense of higher memory requirements. In particular, the search process is avoided if r's request arrives at a bufferer of the message.

The above discussion is simplified in assuming that p is the only member receiving a remote request. As described in Section 2, when an entire region missed a message, on average λ members will send remote requests to an upstream region. As soon as one of them receives a remote repair, it will multicast the repair in its region.

4 Simulation Results

In this section, we evaluate the performance of our buffer management scheme using simulation. We focus on the behavior of the protocol in a local region. The round trip time between any two members in the region is 10ms. The idle threshold T is set to 40ms (i.e., 4 times the maximum round trip time). For simplicity, we assume that retransmission requests and repairs are not lost.

We first evaluate the effectiveness of our feedback-based short-term buffering scheme in a region with 100 members. We simulate the outcome of an IP multicast by randomly selecting a subset of members to hold a message initially. All other members simultaneously detect the loss and start sending local requests. We measure how long these initial members buffer the message. The result is shown in Figure 5 (note that the x-axis is in logarithmic scale). As can be seen from the figure, the amount of buffering time decreases as the initial IP multicast has reached more members.

In Figure 6 we take a closer look at one of the data points in Figure 5 where one member holds a message initially. We compare the number of members which have received the message with the number of members which buffer the message as error recovery proceeds. As can be seen from the figure, when only a small percentage of members have received the message, almost all of them buffer the message. The number of short-term bufferers decline rapidly when an overwhelming majority of members (96% in this case) have received the message. The results in these two figures demonstrate that our feedback-based scheme is effective in allocating buffer space to those messages most needed in the system.

Next we investigate the penalty in error recovery latency due to a need to search for a bufferer. We assume that a remote request arrives at a randomly chosen mem-

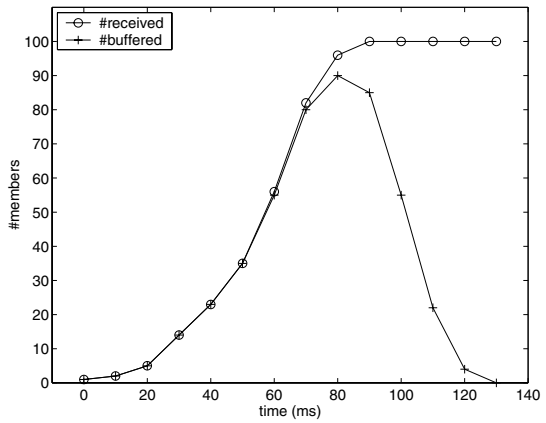

Figure 6. Comparison between the number of members which have received a message and the number of members which buffer the message as error recovery proceeds.

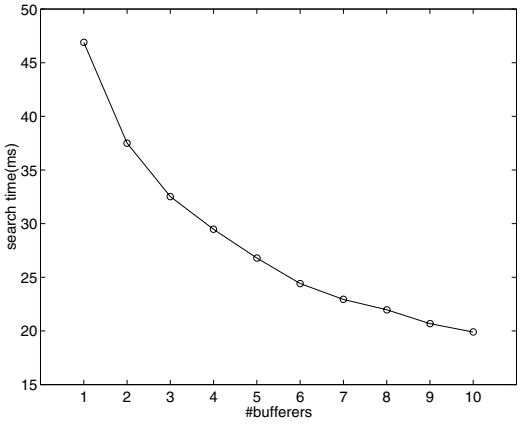

Figure 7. Search time decreases as the number of bufferers increases.

ber in a region with 100 members. The simulation is repeated 100 times with different random seeds and the average is taken. Figure 7 shows that the search time decreases as the number of bufferers increases. (The search time is 0 if the request arrives at a bufferer.) With 10 bufferers, for example, the average search time is 20ms (i.e. twice the round trip time). In a wide area network, the latency between two regions is usually much higher than the latency within a region. Hence the search time is likely to be a small fraction of the total recovery latency.

In Figure 8 we show how the search time changes when the size of the region increases from 100 members to 1000 members. The number of bufferers is fixed at 10.

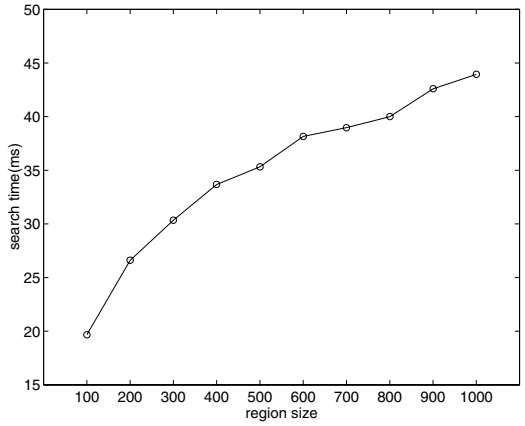

Figure 8. Search time as the size of the region increases.

The figure indicates that the degree of increase in search time is much smaller than that in region size: when the region size increases by a factor of 10, the corresponding search time only increases by a factor of 2.2. With 1000 members, the percentage of bufferers is only 1%. Compared with the case where every member buffers the message, our algorithm reduces the amount of buffer space by a factor of 100.

5 Experimental Results

In this section, we compare the amount of buffer requirements in Bimodal Multicast with that in RRMP on the UNIX platform. The experiment was conducted in a group of 30 machines in a local area network. The sender sends $1K$ byte messages at a rate of 100 messages per second. Messages are delivered to the application in FIFO order. We randomly drop messages with probability 1% at each receiver and compare the number of messages a receiver keeps in its buffer between the two protocols. The results are shown in Figure 9. The x-axis indicates the times when the measurements were taken and the y-axis indicates the number of buffered messages.

Recall that a receiver in the Bimodal Multicast protocol buffers received messages for a fixed amount of time after their initial reception and then garbage collects the message [2]. In the current implementation, the length of a gossip round is 100ms and a receiver keeps a message for 10 rounds. The figure shows that the number of messages in a member's buffer is around 100. In contrast, the RRMP protocol divides its buffer space into two parts: a short-term buffer and a long-term buffer. When a member first receives a message, it keeps

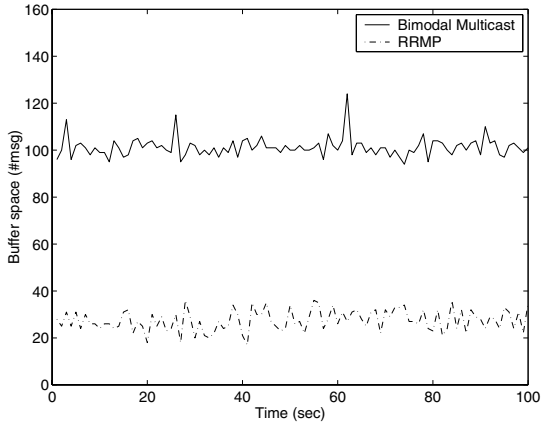

Figure 9. Comparison of buffer requirements between Bimodal Multicast and RRMP in a group of 30 members in a local area network.

the message in its short-term buffer until no request for this message has been received for a certain period of time (50ms in the current implementation). Then the member makes a random choice to become a long-term bufferer with probability C/n. In this experiment, we set $C = 6$ and $n = 30$. Hence on average 20% of the members in a region serve as long-term bufferers. A long-term bufferer keeps the message for 1 second. The figure shows that the resulting buffer requirements are substantially smaller than that for the Bimodal Multicast protocol.

The amount of buffer space in Bimodal Multicast can be reduced if a member buffers received messages for a shorter period of time. In order to be comparable to RRMP, a member should buffer a message for approximately 250ms. However, we have shown in [15] that a noticeable fraction of message losses in Bimodal Multicast may take longer than 250ms to recover due to randomization. If a message loss cannot be recovered after a certain amount of time, the protocol gives up on the message and reports the loss to the application. Consequently, the application may experience a higher loss rate if the amount of buffering time is reduced.

One concern with the two-phase buffering scheme in the RRMP protocol is its potential negative impact on error recovery latency: after a message has become idle (i.e. no request for this message has been received for 50ms), only a subset of members in a local region will continue to buffer the message. If a member discards a message and then later receives a retransmission request for that message, it cannot answer the request itself and needs to search for a bufferer of the message. This is usually not a problem when all members are in a local area network, or when message losses occur randomly and independently, because our previous work has demonstrated that error recovery latency in this case is much smaller than 50ms [15].

The situation is quite different in a wide area network where all members in a region may miss the same message. To study the behavior of the RRMP protocol under such situations, we conduct another experiment where we emulate a wide area network by dividing the 30 members evenly into two local regions. The sender is in one of the regions. Messages sent within a local region experience the normal delay of the underlying physical network. Messages sent between the two regions have an additional delay of 30ms and a random loss probability of 5% to emulate wide area links. (No loss is introduced on messages sent within a local region.) Because all members in the downstream region will miss the same message, the lost message can only be repaired through the remote recovery phase. Due to the long latency between the two regions (the round trip time is larger than 50ms), a member in the sender's region may receive a remote request from a downstream member, asking for a message that it has already discarded. In this case it needs to search for a bufferer of the message.

As a target for comparison, we implemented a *single-phase* buffering scheme in which all members continue to buffer an idle message for 1 second (i.e. every member is a long-term bufferer). We compare the error recovery latency between the two schemes and show the results in Figure 10. The x-axis is the error recovery latency in milliseconds and the y-axis is the percentage of message losses that are recovered within the corresponding amount of time. The figure indicates that the two-phase buffering scheme incurs only a small performance penalty in error recovery latency while providing a substantial reduction in buffer requirements.

6 Discussion

In RRMP, a member may discard a message before the message has been received by all members in the group. This is in contrast to stability detection protocols where a message is discarded only after it has been delivered everywhere. Consequently, our buffering scheme introduces a small probability of violating the reliability guarantee of the multicast service. Such probability can be made arbitrarily small with carefully chosen parameters for the protocol, but still must be accounted for when designing an application.

Applications that require a stronger guarantee should use a protocol that provides better reliability, such as vir-

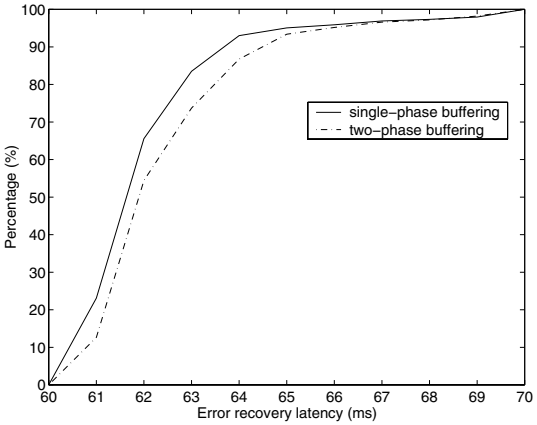

Figure 10. Comparison of error recovery latency with two buffering schemes.

tual synchrony [1]. The probabilistic guarantees offered by RRMP have the benefit of superior scalability and intrinsic robustness in networks subject to message loss and process failures, but are not appropriate when absolute guarantees of reliability are needed.

7 Related Work

In the RRMP protocol, the set of long-term bufferers are chosen randomly from all receivers in a region. Previously we proposed a deterministic algorithm [12] that chooses a subset of receivers in a group to serve as bufferers using a hash function as described in Section 1. It is interesting to compare these two approaches.

We believe that the choice reflects a trade-off between network traffic and computation overhead. Under the deterministic algorithm, a receiver can find out the set of bufferers for a message by applying the hash function to the network address of each member in its region. This avoids the latency and network traffic associated with the search process. However, it incurs certain computation overhead because the hash function needs to be calculated each time a message is received. In [12] van Renesse proposed the design of an efficient hash function.

One advantage of the randomized algorithm is that it allows easy adaptation to group membership dynamics: when a receiver voluntarily leaves the group, it can transfer messages in its long-term buffer to randomly selected receivers in its region. It is not clear how this can be done with a deterministic algorithm.

lpbcast is a gossip-based message dissemination protocol that has been used to implement a publish/subscribe system [6]. It uses a scalable membership management algorithm where each member maintains membership information for only a random subset of members in the group. In addition, the protocol uses an age-based garbage collection scheme to purge old messages from the system [10]. More specifically, a member in this protocol associates an age with each message in its buffer. The age is initialized to 0 when the message is first received and is updated in each gossip round to reflect the amount of time the message has spent in the system. A member discards messages with a high age when its buffer is full.

A common goal of both *lpbcast* and RRMP is to allocate buffer space to useful messages in the system. However, the two protocols are different in significant ways. In *lpbcast*, a member sends a gossip message to some randomly selected destinations periodically. Messages that have been gossiped for a long time tend to be delivered by many processes. Such messages are considered less useful to buffer than recently published messages. In contrast, a member in RRMP sends retransmission requests to randomly selected members upon detection of a message loss. Messages that have been requested recently are likely to be needed by other members. Such messages are buffered for a longer period of time under our feedback-based buffering scheme. Moreover, the protocol employs a two-phase buffering algorithm that scales well in a heterogenous network.

8 Conclusion

Designing an efficient buffer management algorithm is challenging in large multicast groups where no member has complete group membership information and the delivery latency to different members could differ by orders of magnitude. This paper has presented an innovative two-phase buffering algorithm that explicitly addresses variations in delivery latency seen in large multicast groups. Unlike tree-based protocols where a repair server bears the entire burden of buffering messages for a local region, RRMP achieves better load balancing by spreading the load of buffering among all members in the region. Compared with stability detection protocols, our buffering algorithm has low traffic overhead because it does not require periodic exchange of message history information among members in the group. Simulation and experimental results demonstrate that the algorithm has good performance.

Although we present our buffer optimization in the framework of the RRMP protocol, similar techniques can be applied to other reliable multicast protocols as well. In the following we summarize the main ideas in our algorithm and discuss how they can be applied to the

SRM protocol:

- The dissemination status of the initial IP multicast for each message can be different. A good buffering algorithm should adaptively allocate buffer space to messages most needed in the system.
- Retransmission requests can be used as feedback to estimate the dissemination status of a multicast message. In the SRM protocol, retransmission requests and replies are multicast to the entire group. If a receiver has not received any request for a message after a sufficiently long period of time, it can conclude that the message is stable. Such information can be helpful to the application in managing its buffer space.
- In a large multicast group, it may take a long time for a message to become stable. While research on stability detection focuses on optimizing buffer space after a message has become stable, our work aims to reduce buffer space even before stability has been achieved. In the context of SRM, instead of having every receiver buffer a message until the message is stable, a randomly selected subset of receivers can serve as bufferers for the message.
- In a wide area network, the latency between two regions can be much higher than the latency within a region. Our buffering algorithm addresses this difference in latency by dividing buffer space into two parts: the short-term buffer allows a local loss to be recovered quickly within the local region, while the long-term buffer serves to satisfy remote requests from downstream regions without consuming too much buffer space. Although the original SRM protocol is unstructured, extensions have been made to introduce an error recovery hierarchy into the protocol [11, 14]. Our two-phase buffering scheme can be applied in such a hierarchy.

Acknowledgments

We would like to thank Fred Douglis and the anonymous reviewers for comments on an early draft of the paper.

References

[1] K. P. Birman. *Building Secure and Reliable Network Applications*. Manning Publishing Company and Prentice Hall, 1997.

[2] K. P. Birman, M. Hayden, O. Ozkasap, Z. Xiao, M. Budiu, and Y. Minsky. Bimodal multicast. In *ACM Transactions on Computer Systems*, May 1999.

[3] D. D. Clark and D. L. Tennenhouse. Architectural considerations for a new generation of protocols. In *Proceedings of ACM SIGCOMM*, 1990.

[4] S. E. Deering and D. R. Cheriton. Multicast routing in datagram internetworks and extended LANs. In *ACM Transactions on Computer Systems*, May 1990.

[5] R. Durrett. *The Essentials of Probability*. Duxbury Press, 1994.

[6] P. T. Eugster, R. Guerraoui, S. B. Handurukande, A. M. Kermarrec, and P. Kouznetsov. Lightweight probabilistic broadcast. In *Proceedings of the International Conference on Dependable Systems and Networks*, July 2001.

[7] S. Floyd, V. Jacobson, S. McCanne, C.-G. Liu, and L. Zhang. A reliable multicast framework for lightweight sessions and application level framing. In *Proceedings of ACM SIGCOMM*, 1995.

[8] K. Guo and I. Rhee. Message stability detection for reliable multicast. In *Proceedings of IEEE INFOCOM*, 2000.

[9] H. Holbrook, S. Singhal, and D. Cheriton. Log-based receiver-reliable multicast for distributed interactive simulation. In *Proceedings of ACM SIGCOMM*, 1995.

[10] P. Kouznetsov, R. Guerraoui, S. B. Handurukande, and A. M. Kermarrec. Reducing noise in gossip-based reliable broadcast. In *Proceedings of the 20th IEEE Symposium on Reliable Distributed Systems*, Oct. 2001.

[11] C.-G. Liu, D. Estrin, S. Shenker, and L. Zhang. Local error recovery in SRM: Comparison of two approaches. In *IEEE/ACM Transactions on Networking*, Dec. 1998.

[12] O. Ozkasap, R. van Renesse, K. P. Birman, and Z. Xiao. Efficient buffering in reliable multicast protocols. In *International Workshop on Networked Group Communication*, Nov. 1999.

[13] S. Paul, K. Sabnani, J. Lin, and S. Bhattacharyya. Reliable multicast transport protocol (RMTP). In *IEEE Journal on Selected Areas in Communication, special issue on Network Support for Multipoint Communication*, 1997.

[14] P. Sharma, D. Estrin, S. Floyd, and L. Zhang. Scalable session messages in SRM using self-configuration. Technical report, University of Southern California, 1998.

[15] Z. Xiao. *Efficient Error Recovery for Reliable Multicast*. PhD thesis, Cornell University, Jan. 2001.

[16] Z. Xiao and K. P. Birman. A randomized error recovery algorithm for reliable multicast. In *Proceedings of IEEE INFOCOM*, Apr. 2001.

[17] R. Yavatkar, J. Griffioen, and M. Sudan. A reliable dissemination protocol for interactive collaborative applications. In *Proceedings of ACM Multimedia*, 1995.