A Comparison of Server–Based and Receiver–Based Local Recovery Approaches for Scalable Reliable Multicast*

Sneha K. Kasera, Jim Kurose and Don Towsley
Department of Computer Science
University of Massachusetts
Amherst, MA 01003
{kasera,kurose,towsley}@cs.umass.edu

Abstract

Local recovery approaches for reliable multicast have the potential to provide significant performance gains in terms of reduced bandwidth and delay, and higher system throughput. In this paper we examine two local recovery approaches — one server–based, and the other receiver–based, and compare their performance. The server–based approach makes use of specially designated hosts, called repair servers, co–located with routers inside the network. In the receiver-based approach, only the end hosts (sender and receivers) are involved in error recovery. Using analytical models, we first show that the two local recovery approaches yield significantly higher protocol throughput and lower bandwidth usage than an approach that does not use local recovery. Next, we demonstrate that server–based local recovery yields higher protocol throughput and lower bandwidth usage than receiver–based local recovery when the repair servers have processing power slightly higher than that of a receiver and several hundred kilobytes of buffer per multicast session.

1 Introduction

Reliable multicast is required by many applications such as multicast file transfer, shared whiteboard, distributed interactive simulation and distributed computing. These applications can potentially have several thousands of participants scattered over a wide area network. Even though current multicast networks provide efficient routing and delivery of packets to groups of receivers, they are lossy and do not provide the reliability needed by the above applications. Designing scalable approaches and architectures for reliable multicast that make efficient use of both the network and end–host resources is a challenging task.

Local recovery approaches for reliable multicast, in which a network entity other than the sender aids in error recovery, have the potential to provide significant performance gains in terms of reduced bandwidth and delay, and higher system throughput. In this paper, we examine and compare two different local recovery approaches, one requiring additional processing and caching inside the network and the other based only on end–to–end means. The first approach, termed server–based local recovery, makes use of specially designated hosts, called *repair servers*, co–located with routers inside the network. Each repair server serves the retransmission requests of receivers in its local area by transmitting local repairs. The repair servers themselves recover lost packets from either higher level repair servers or the sender. Co–locating repair servers with routers is functionally equivalent to placing repair services at the routers and can be accomplished using *active networking* mechanisms [10].

In the second approach, termed receiver–based local recovery, there are no designated repair servers; only the end hosts (sender and receivers) are involved in error recovery. This approach dynamically selects a receiver within a local geographic area to supply the repair, whenever local recovery of a lost packet is attempted.

We analyze the protocol throughput and the network bandwidth usage of the above two local recovery approaches in the face of spatially correlated loss. We first show that local recovery yields higher protocol throughput and lower bandwidth usage than approaches that do not use local recovery. Next we compare the performance of the two local recovery approaches. We demonstrate that server–based local recovery yields higher protocol throughput and lower bandwidth usage than receiver–based local recovery when the repair servers have processing power slightly higher than that of a receiver and several hundred kilobytes of buffer per multicast session.

The remainder of this paper is structured as follows. In Section 2 we examine the existing work on local recovery. In Section 3 we present the two local recovery approaches. Section 4 contains our system model. Section 5 contains the throughput and bandwidth analysis of the two local recovery approaches. We compare the performance of the server–based and the receiver–based approaches in Section 6. In Section 7 we size the repair servers. Conclusions and directions for future work are contained in Section 8.

2 Related Work

Many researchers have recently proposed different approaches for local recovery. These include SRM (with local recovery enhancements) [2], LGC [3],

*This work was supported by ARPA under contract N6601-97-C-8513.

TMTP [14], LORAX [7], LBRM [4] and RMTP [8] and can be broadly classified as follows. LBRM and RMTP are server–based, SRM with local recovery enhancements is receiver–based with randomized local recovery, and TMTP and LORAX are receiver–based with logical tree–based recovery. LGC is a hybrid of the logical tree–based approach and SRM. Our focus, in this paper, is on the first two approaches.

Our work is the first to analytically compare the throughput and bandwidth usage of the repair server (or designated host)–based and receiver–based local recovery approaches. Previous throughput analyses have focused on comparing sender–oriented versus receiver–oriented protocols that do not use local recovery [9, 11, 5] and comparison of logical tree–based local recovery with other non–local recovery approaches [7]. In our work, we consider a richer loss model [13] and also look at bandwidth usage besides processing costs.

3 Protocol Descriptions

We present server–based and receiver–based approaches to local recovery for reliable transmission of data from a sender to multiple receivers. Both approaches are NAK–based. The server–based approach places much of the burden for ensuring error recovery on repair servers placed inside the network. In the receiver–based approach, the error recovery burden is shared among the sender and the participating receivers. In this section we assume that mechanisms are available for scoping multicasts within a "local" region. Some mechanisms for accomplishing this are listed in [6].

3.1 A Server–Based Local Recovery Protocol

We now describe a generic server–based reliable multicast local recovery protocol, L1, that assumes the presence of a *repair* tree. This repair tree is the physical multicast routing tree constructed by the routing protocols with repair servers co–located with routers[1]. The receivers recover lost packets from repair servers and repair servers recover lost packets from either upper level repair servers or the sender. Protocol L1 exhibits the following behavior:

- The sender multicasts packets to a multicast address A that is subscribed to by all receivers and repair servers.
- On detecting a loss, a receiver, after waiting for a random amount of time, multicasts a NAK to the receivers in its neighborhood and the repair server (for the purpose of NAK suppression among receivers inside the neighborhood), and starts a NAK retransmission timer. If the receiver receives a NAK from another receiver for this packet, it suppresses its own NAK and sets the NAK retransmission timer as if it had sent the NAK.
- On detecting a loss, a repair server, after waiting for a random amount of time, multicasts a NAK to the sender (or its upstream repair server) and the other repair servers at the same level in the tree (for the purpose of NAK suppression among repair servers) and starts a NAK retransmission timer. While waiting to send out a NAK for the lost packet, if the repair server receives a NAK for the same packet from another repair server, then it suppresses its own NAK and sets the NAK retransmission timer as if it had sent the NAK.
- If a repair server has the packet for which it received a NAK from a member of the group for which it is responsible, it multicasts the packet to the group. Otherwise, it starts the process of obtaining the packet from the sender (or its upstream repair server) if it has not already done so (as described above).
- The sender, on receiving a NAK from the repair servers, remulticasts the requested packet to all the receivers and repair servers.
- The expiration of the NAK retransmission timer at a repair server (or a receiver) without prior reception of the corresponding packet serves as the detection of a lost packet for the repair server (or the receiver) and triggers the retransmission of a NAK.

We end this section by distinguishing between a log service and a repair service. A log service, as mentioned in [4], provides secondary storage for packets transmitted from the sender. The entire data is stored at the log servers for repairs and "late–comers." Log service should be distinguished from the repair service, particularly if the repair servers are considered network resources that will be shared over several multicast sessions. Logging entire multicast sessions at these repair servers may not scale with the number of multicast sessions because of the storage requirement. In our work, we consider storing only the "recent" data at the repair servers for the sole purpose of providing quick repairs.

3.2 A Receiver–Based Local Recovery Protocol

We now describe a receiver–based protocol for local recovery, that does not assume the presence of repair servers in the network. This protocol, termed L2, is a generic version of SRM [2] with local recovery enhancements. In L2, loss recovery is performed at two levels. At the first level, a receiver tries to recover packets locally from within its local neighborhood. A receiver's local neighborhood consists of all the receivers in the subtree (of the multicast routing tree) emanating from its nearest backbone router at the edge of the backbone[2]. If the receiver is unsuccessful in recovering the packet locally, it tries to recover packets from the sender. We refer to this as "global recovery." Each receiver alternates between local and global recovery until it receives the missing packet. Protocol L2 exhibits the following behavior

[1] This repair tree is different from the logical repair tree constructed, on top of the multicast routing tree, by the reliable multicast approaches described in [7, 14].

[2] This definition and other definitions of local neighborhood can be found in [2].

- the sender multicasts all packets on address A which is subscribed by all the participating receivers.
- on detecting a loss, a receiver waits for a random amount of time and multicasts a local NAK addressed only to receivers in its neighborhood and starts a local NAK retransmission timer (first–level recovery); if the receiver does not receive the lost packet before the local NAK retransmission timer expires, it waits a random amount of time, multicasts a global NAK to address A and starts a global NAK retransmission timer (global recovery). If the receiver does not recover the lost packet before the global NAK retransmission timer expires, it restarts the local recovery.
- on receiving a local NAK for which it has the associated packet, a receiver multicasts the packet only to its neighborhood; before sending this repair packet, a receiver waits for a certain random amount of time and suppresses its own transmission if another receiver sends out the repair packet in this time. On receiving a global NAK for a packet, a receiver suppresses its own NAK if there is any.
- upon receiving a global NAK, the sender multicasts the packet to A.

The above protocol attempts to ensure that only one NAK and only one repair are generated at either recovery level. Unlike SRM [2], L2 generates *global repairs only from the sender* thereby avoiding the risk of multiple global repairs. In SRM, receivers and the sender use a random back-off strategy to generate a repair. Here all receivers and the sender " listen" for a repair. If they do not receive the repair within a certain time then they generate a repair if they have it. Note that a repair has to be processed at each node whether it is sent or received. Hence the overhead of processing a global repair under SRM and L2 will be comparable. For local recovery, no particular host is likely to have the repairs and hence, as in SRM, L2 uses a random back-off strategy for generating local repairs. As described above, protocol L2 is easily generalized to multiple levels of recovery.

4 System Model

In this section we present the network loss and system model that will be used for the performance analysis of protocols L1 and L2. In [13] Yajnik *et al* have observed that most losses take place in the links from the source site to the backbone (we call this the source link) and in the links (called "tail links") from the backbone to the individual sites (termed stub domains in [1]), as shown in Figure 1. The backbone and the individual sites have been observed to be mostly loss free. It has also been noted in [4] that tail links are likely to be bottlenecks for the foreseeable future. Hence in our loss model we consider loss only at the source and tail links. We also assume that loss events are temporally independent. Let the loss probability in the source link and each tail link be p_l. The end–to–end loss probability as seen by a receiver, p, is equal to $1-(1-p_l)^2$. We also assume that NAKs are never lost. This assumption could be relaxed by following the analysis given in [12].

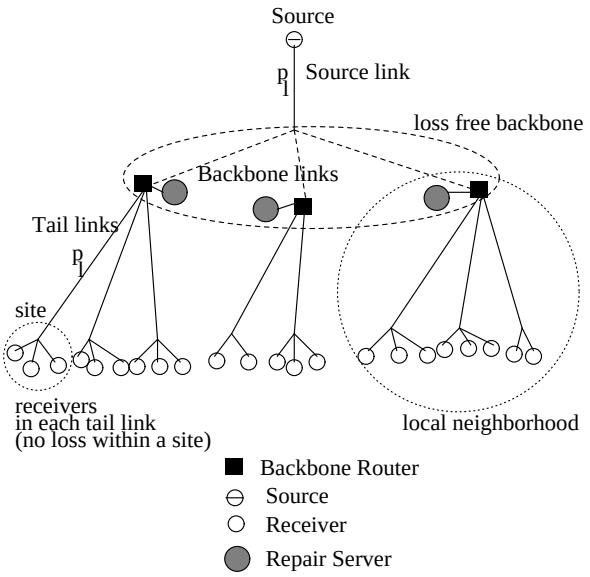

Figure 1: System Model

Let there be one source link and T tail links where each tail link has several receivers. Let each upstream backbone router at the edge of the backbone have k tail links. The number of such backbone routers, W is equal to T/k. We assume that communication between two receivers on different tail links takes place only through a backbone router(s). Direct site-site links (termed stub-stub edges in [1]) are likely to be very rare.

The above loss model has implications on the placement of repair servers inside the network. To be able to take advantage of local recovery, a repair server should be both upstream of and as close to the point of loss as possible, hence they should be placed with routers at the edge of the backbone. Figure 1 shows the placement of repair servers.

We end this section by noting that for the system model two–level recovery is sufficient for protocol L1. Also, the multicast address for NAK suppression among receivers in L1 is chosen such that only receivers within a tail participate in NAK suppression. Since receivers within a tail see the same loss, each receiver processes a NAK only when it losses the corresponding packet. Each site sends only one NAK to the repair server per loss.

5 Performance Analysis

In this section we analyze the end system processing and network bandwidth requirements of protocol L2. The analysis of L1 is simple (see [6]), and is omitted here for the sake of brevity.

Following the approach first suggested in [9], the receive processing cost (time) is determined by computing the total amount of processing involved in *correctly* receiving a randomly chosen packet. This in-

cludes the time required to receive those copies of this packet (i.e., the original copy plus any retransmissions) that arrive at the receiver, the time required to send/receive any NAKs associated with this packet and the time needed to handle any timer interrupts associated with this packet. The send processing cost is determined by the total sender processing involved in correctly transmitting a packet correctly to all the receivers. This includes the cost required to process any received NAKs, and process retransmissions that are sent out in response to these NAKs.

X_p	–	time to process the transmission of a packet
X_n	–	time to process a NAK at a sender
Y_p	–	time to process a newly received packet
Y_t	–	time to process a timeout
Y_n	–	time to process a NAK at a receiver
p	–	end–to–end loss probability at a receiver
p_l	–	source link or tail link loss probability
T	–	total no. of tail links
k	–	number of tail links per backbone router
M	–	number of transmissions from the sender for all receivers to receive the packet correctly if there were no local recovery
M_n	–	number of transmissions from the sender for all receivers in a local neighborhood to receive the packet correctly if there were no local recovery
$M^{'}$	–	number of transmissions from the sender to ensure that at least one receiver in each neighborhood receives the packet correctly
$M_n^{'}$	–	number of transmissions from the sender to ensure that at least one receiver in a neighborhood receives the packet correctly if there were no local recovery
M_t	–	number of transmissions (global or local) required for any receiver r to correctly receive a packet
X^ω,Y^ω	–	send and receive per packet processing time in protocol $\omega \epsilon$ {L1, L2}

Table 1: Notation

Our bandwidth measure is *total bandwidth usage*[3] defined as the total number of bytes sent per successful packet transmission along all of the links to all the receivers.

5.1 Processing Cost Analysis

We now derive expressions for sender and receiver processing requirements for protocol L2. Table 1 describes the notation used in the analysis, some of which has been reintroduced from [11, 5]. We assume that processing times are arbitrarily distributed and independent of each other. We assume that the required processing times required to send and receive a packet are the same (it has been observed in [5] that they are almost the same). We also make the optimistic assumption that only one retransmission is generated per NAK during local recovery in L2. This assumption will result in lower receiver processing costs for L2. The mean processing time at the sender for a randomly chosen packet is

$$E[X^{L2}] = (E[M^{'}] + E[\lfloor (M - M^{'})/2 \rfloor])E[X_p] + (E[M^{'}] + E[\lfloor (M - M^{'})/2 \rfloor] - 1)E[X_n] \quad (1)$$

The first term corresponds to processing required for transmissions and retransmissions of the packet. This term is made up of two parts. The first part is the expected processing cost of (re)transmissions of the packet so that at least one receiver in every neighborhood receives the packet. The second part is the expected processing cost of additional retransmissions from the sender when receivers cannot locally recover the lost packet even though it is available in the local neighborhood. Recall that a receiver alternates between local and global recovery in that order. Therefore, $\lfloor (M - M^{'})/2 \rfloor$ is the number of additional transmissions required from the sender to ensure that all the receivers correctly receive the packet. The second term corresponds to the processing of global NAKs received by the sender. The number of global NAKs is equal to one less than the number of (re)transmissions from the sender because only one NAK is generated per loss due to NAK suppression[4].

The mean processing time at the receiver for a randomly chosen packet is

$$\begin{aligned} E[Y^{L2}] &= (E[M^{'}] + E[\lfloor (M - M^{'})/2 \rfloor])(1 - p_l)^2 E[Y_p] \\ &+ (E[\lceil (M_n - M_n^{'})/2 \rceil)(1/k + (k-1)(1-p_l)^2/k)E[Y_p] \\ &+ (E[M^{'}] + E[\lfloor (M - M^{'})/2 \rfloor] - 1)E[Y_n] \\ &+ (E[M_n^{'}] + E[\lceil (M_n - M_n^{'})/2 \rceil] - 1)E[Y_n] \\ &+ (E[(M_t - 2)^+] + E[(M_n^{'} - 2)^+])E[Y_t] \end{aligned} \quad (2)$$

The first two terms correspond to the processing of global and local transmissions, respectively, at the receiver. The third term in (2) corresponds to the mean processing time required to send and receive global NAKs. The fourth term is the mean processing time required for sending and receiving local NAKs. It is important to note here that $M_n^{'}$ local NAKs are generated even before the packet is available in the local neighborhood. The last term is the mean processing time required for timer routine executions. Here, $(x)^+ = \max\{0, x\}$. Each term has been explained in detail in [6]. Now, both (1) and (2) contain the term $E[\lfloor (M - M^{'})/2 \rfloor]$ and (2) contains the term $E[\lceil (M_n - M_n^{'})/2 \rceil]$. Since random variables M and $M^{'}$

[3] In [6] we have also looked at another bandwidth metric.

[4] Here the assumption is that NAK suppression works perfectly.

(and also M_n and $M_n^{'}$) are not independent it is not easy to obtain simple expressions for these expressions. We bound $E[\lfloor (M - M^{'})/2 \rfloor]$ and $E[\lceil (M_n - M_n^{'})/2 \rceil]$ by

$$\begin{aligned} E[\lfloor (M - M^{'})/2 \rfloor] &> (E[M] - E[M^{'}])/2 - 1/2 \\ E[\lceil (M_n - M_n^{'})/2 \rceil] &> (E[M_n] - E[M_n^{'}])/2 \end{aligned} \quad (3)$$

which is expected to be tight when the loss probability is high and number of tail links is large. The rest of the expressions required for evaluating the terms of equations (1) and (2) are available in [6].

The overall protocol throughput for L2 is given by the minimum of the per–packet processing rates at the sender, $\Lambda_s^{L2} = 1/E[X^{L2}]$ and the receiver, $\Lambda_r^{L2} = 1/E[Y^{L2}]$

$$\Lambda_o^{L2} = \min\{\Lambda_s^{L2}, \Lambda_r^{L2}\} \quad (4)$$

A similar analysis is done (see [6]) to obtain the processing rates at the sender, receiver and repair server for protocol L1. Under the assumption that the repair server is never a bottleneck, the overall protocol throughput for L1 is given by the minimum of the per–packet processing rates at the sender and the receiver.

$$\Lambda_o^{L1} = \min\{\Lambda_s^{L1}, \Lambda_r^{L1}\} \quad (5)$$

5.2 Bandwidth Analysis

To analyze the bandwidth performance we consider the network to be made up of three types of links, a source link, backbone links and tail links. Then the mean total bandwidth usage, denoted by B^{ω}, where $\omega \epsilon \{L1, L2\}$, can be expressed as,

$$E[B^{\omega}] = E[B_s^{\omega}] + E[B_b^{\omega}]W + E[B_t^{\omega}]T \quad (6)$$

Here $E[B_s^{\omega}]$, $E[B_b^{\omega}]$ and $E[B_t^{\omega}]$ are the mean bandwidth usage on the source link, a backbone link and a tail link respectively, for protocol ω. For determining these quantities we need to find the mean number of packets flowing in each of the links per successful transmission of a packet from the sender to all the receivers. For simplicity, we assume that the source, backbone and tail links each consists of a single physical link. A packet is counted if it is to be offered to a link.

The expressions for $E[B_s^{\omega}]$, $E[B_b^{\omega}]$ and $E[B_t^{\omega}]$ where $\omega \epsilon \{L1, L2\}$, can be found in [6].

6 Throughput and Bandwidth Comparisons

In this section we first compare the throughput and bandwidth usage of protocols L1 and L2 with a protocol N2 [11] that does not use local recovery to establish the benefits of local recovery. N2 is a receiver oriented protocol that uses global NAK suppression and is shown to have the highest protocol throughput among global recovery schemes [11]. Next we compare the performances of L1 and L2. In computing L1's throughput we assume that the repair server has sufficient processing power and is never a bottleneck.

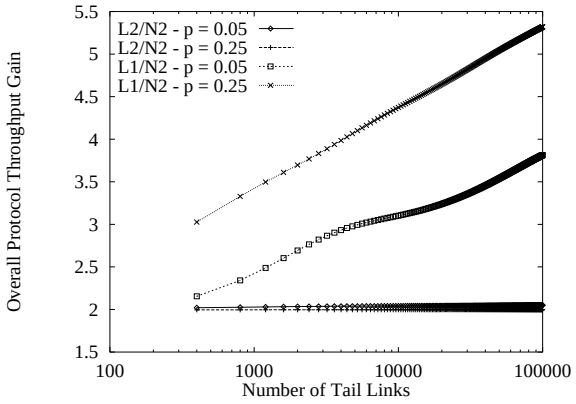

Figure 2: Throughput Gain

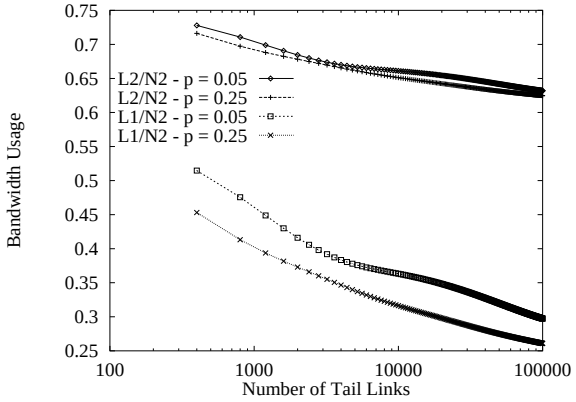

Figure 3: Bandwidth Reduction

The processing power required for this to be true is determined in the next section.

In order to compute the send and receive processing costs we use the measurements reported in [5] for the processing times associated with sending/receiving a data packet and a NAK packet, and processing timeouts. We use $E[X_p] = E[Y_p] = 500\mu$secs, $E[X_n] = E[Y_n] = 85\mu$secs and $E[Y_t] = 32\mu$secs. Here a data packet is of size 1024 bytes and a NAK packet is of size 32 bytes. The number of tails per neighborhood (as defined in Section 3.2), k, and the number of tails per repair server, are both set to 8.

Figures 2 and 3 show the higher throughput and bandwidth reduction due to local recovery. The performance of L1 is significantly better than N2. Further, this behavior becomes more pronounced as the number of tails, T, and the loss probability, p increase. L2 also performs much better than N2. However, the ratio of the throughput obtained under L2 and N2 remains equal to 2 and does not change significantly with an increase in T or p. There is a slight reduction in the ratio of bandwidth usage obtained under L2 and N2 as the number of tails increases. This is because fewer packets flow in the network under L2 in comparison to N2.

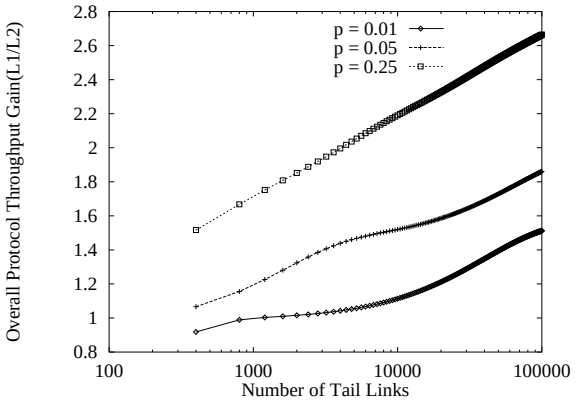

Figure 4: Throughput Gain (L1/L2)

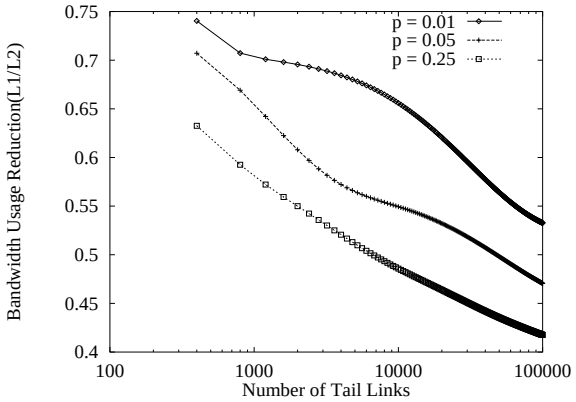

Figure 5: Bandwidth Reduction (L1/L2)

Figure 4 shows how the ratio of the throughputs of protocols L1 and L2 varies with T and p. In Figure 5, we compare the bandwidth usage of L1 and L2. One observes that L1 has a much higher throughput and uses much less bandwidth in comparison to L2. This behavior becomes more prominent as the number of tails T and the loss probability p increase. The L1 sender is responsible for correctly transmitting packets only over the source link. In the case of L1 the receivers multicast NAKs only within their tail and locally recover lost packets from the repair server. They do not receive any unwanted global retransmissions or unwanted local or global NAKs. L2 on the other hand requires a sender to supply repairs even if they are available in the local neighborhoods due to failure of local recovery because of losses on tails inside the neighborhoods. L2 also requires local NAK suppression within the entire local neighborhood during the generation of local NAKs and also global NAK suppression in the entire network during the generation of global NAKs. For these reasons the sender and the receivers in the case of L2 do much more processing in comparison to the sender and receivers under L1 leading to higher sender and receiver throughput under L1. This results in a higher overall protocol throughput under L1. There appears to be an exception when $p = 0.01$ and $T < 1000$. However, this is due to the fact that the bounds used in the analysis of L2 (equation (3)) are loose for small p and T and favor L2. Elementary sample path analysis shows L1 to be always better than L2.

As more data packets and NAKs are multicast to the entire network under L2, L1 uses much less bandwidth than L2.

So far we have not considered the processing costs at the repair servers. To account for these processing costs let us assume that the repair servers have the same processing power as the sender and receivers. Hence we consider $E[X_p]$, $E[Y_p]$, $E[X_n]$, $E[Y_n]$, $E[Y_t]$ for a repair server to be the same as that for a receiver or sender. We then find the mean total per packet processing costs at all nodes (sender, repair servers, receivers) under protocol L1 (denoted by s_p^{L1}). We also

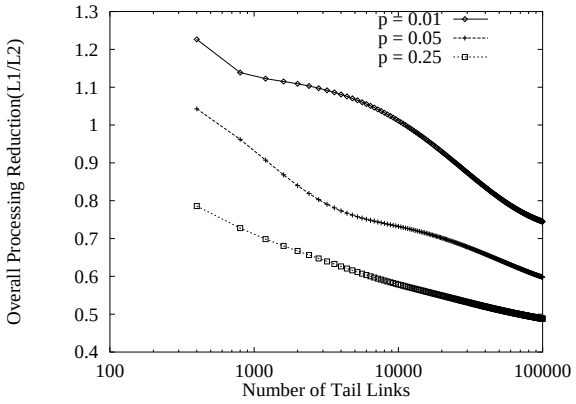

Figure 6: Mean Total Per Packet Processing Ratio (L1/L2)

find the mean total per packet processing costs at all nodes (sender, receivers) under L2 (denoted by s_p^{L2}). For both s_p^{L1} and s_p^{L2}, we assume that there is only one receiver per tail link. Figure 6 shows how s_p^{L1}/s_p^{L2} varies with T and p. It can be seen for $p = 0.01$ and $T < 10000$, the mean total per packet processing cost under L1 is more than that of L2. This is because for low loss probabilities fewer packets are lost and retransmitted but the repair server still does the extra work of receiving all of the packets. With the increase in the number of tails and loss probability the mean total per packet processing cost under L1 decreases relative to that under L2. Noting the fact that the receiver processing cost under L1 is less than that under L2, adding more receivers per tail will further reduce, relatively, the mean total per packet processing cost under L1.

In summary, we see that local recovery leads to higher protocol throughput and lower bandwidth usage. While comparing two local recovery protocols L1 and L2 we note that L1 has higher sender and receiver throughput and uses much less bandwidth. If repair servers are not bottlenecks then we see a marked im-

provement in overall protocol throughput under protocol L1 in comparison to that obtained under protocol L2. Interestingly, if we consider the repair servers to have the same processing power as the sender and receivers then there is an overall reduction in processing costs in the repair server based approach unless the network is small or losses are low.

In [7], Levine *et al* have analyzed a receiver based local recovery approach that constructs a logical tree with only the sender (as root) and the receivers on top of a multicast routing tree. Instead of special repair server hosts, the non–leaf receivers take up the task of providing repair service to its children. The performance of this approach is very sensitive to the branching factor of the logical tree. We extended the analysis in [7] using our loss model and found that the logical tree approach provides excellent throughput and bandwidth performance when the branching factor is very small. However, for a large number of receivers, a smaller branching factor increases the depth of the tree resulting in longer recovery paths. This can potentially lead to high delays. A meaningful comparison of the logical tree–based approach with the two local recovery approaches examined in this paper is possible only by modeling the delay behavior in addition to modeling the throughput and bandwidth usage.

7 Sizing the Repair Servers

In the previous section the throughput of protocol L1 was obtained under the assumption that the repair server has sufficient processing power and is never a bottleneck. In this section we determine the processing and buffer requirements of a repair server so that it does not become a bottleneck.

7.1 Processing Power

A repair server must perform receive–side processing of packets and NAKs, and send–side processing of repairs. Since a receiver performs more processing than the sender under L1, we determine the processing power of the repair server in terms of that receiver (using equations (2) and (3) in [6]). If a repair server can match the receiver throughput then it will never be a bottleneck in a multicast session that uses protocol L1.

In Figure 7 we plot the ratio of the mean repair server processing cost (assuming that it has the same processing power as a receiver) and the mean receiver processing cost as the end-to-end loss probability, p is varied. A repair server needs a processing power that is no more than 1.28 times that of a receiver for a number of tail links per repair server equal to 8 and for loss probabilities less than 0.25.

Thus a repair server needs only be a little faster than a receiver to avoid becoming a bottleneck itself for reasonable loss probabilities. This result is good for one multicast session. If a repair server is to handle K multicast sessions, then it has to be K time faster.

7.2 Buffer Requirements

A repair server has to buffer a set of packets in order to be able to retransmit them due to losses in the tail links. Theoretically, each packet should be held for an infinitely long time to ensure perfectly reliable local recovery from a repair server. Realistically, it is possible to use a finite size buffer such that the probability of the failure of local recovery from the repair server is extremely small. In the rare event of local recovery failure, the lost packet should be retrieved from the sender.

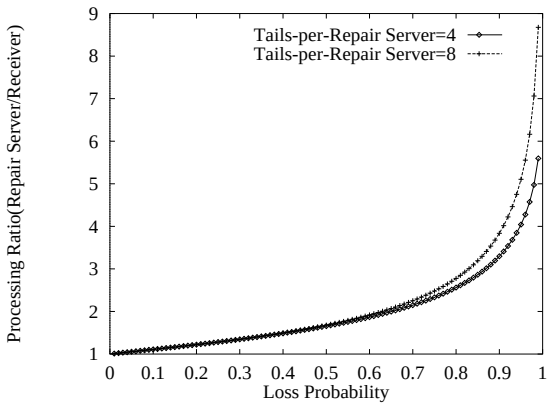

Figure 7: Ratio of Repair Server to Receiver Processing Cost

We have used simple analysis (see [6]) to determine the expected buffer requirement at each repair server, as a function of the probability of local recovery failure(ϵ). The expected buffer requirement depends upon the delay between the repair server and the receivers, the number of tails per repair server and the loss probability. Figure 8 shows an example of the dependence of expected buffer occupancy (in kbytes) on ϵ for two different end–to–end loss probabilities. Here the maximum round trip delay between the repair server and the receivers is set to 20ms and the number of tails per repair server is set to 8.

The expected buffer occupancy at a repair server is 198 kbytes and 137 kbytes, for $p = 0.25$ and $p = 0.05$ respectively when $\epsilon = 10^{-6}$. Thus a repair server requires several 100 kilobytes of buffer space per multicast session for reasonable loss probabilities.

8 Conclusions

We have investigated two local recovery approaches for scalable reliable multicast. In the server–based approach, designated hosts inside the network are used as repair servers. In the receiver–based approaches, error recovery involves only the participating receivers and the sender. Using analysis, we demonstrated the performance gains in using the server–approach in terms of protocol throughput, network bandwidth and overall processing costs. The performance gains increase as the size of the network and the loss probability increase making the server–based approach more scalable with respect to these parameters. We also estimated the processing power and buffer capacity required at the repair servers per multicast session for achieving the performance gains.

A server–based approach could be deployed to improve performance in Intranets where services requir-

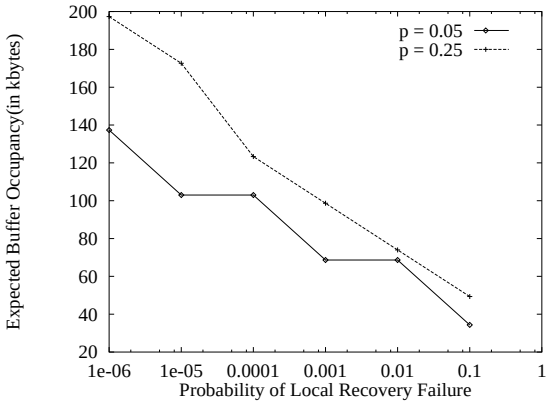

Figure 8: Buffer Requirement at a Repair Server

ing reliable multicast are offered. An Intranet network provider could place the repair servers at appropriate locations. Depending on the processing power and buffer space that can be provided on the repair servers, the network provider could restrict the use of repair servers for only certain number of multicast sessions (by assigning separate multicast addresses and limiting the use of repair servers only for those addresses). As far as the Internet is concerned, wide scale *static* deployment of repair servers with sufficient processing power and buffer capacity may not be easy. It is possible that a more dynamic receiver–based approach, even with a lower performance, may be more attractive. Given the performance improvements of the server–based approach, it is worth studying how to make it more dynamic and flexible. It is also worth exploring the use of repair servers for congestion control.

Future work can proceed in the following directions. The system model could be extended to take into account temporal correlation in the network loss[13] and to also consider more complex topologies with heterogeneity. In determining the buffer requirement at a repair server, we considered failure of local recovery due to limiting the number of retransmissions from the repair server. Local recovery could also fail if we relax the assumption that every packet finds a free buffer on its arrival at the repair server. We plan to extend our analysis to determine the buffer requirement as a function of the probability of local recovery failure due to both these reasons for different buffer management policies. Recently there has been an increasing interest in reliable multicast approaches using forward error correction (FEC). These approaches are based on end-to-end recovery from the sender. They have the potential to reduce network bandwidth usage. It would be interesting to compare the bandwidth usage of local recovery approaches with those that use FEC.

References

[1] K. Calvert, M. Doar and E. Zugera, *Modeling Internet Topology.* IEEE Communications Magazine, June 1997.

[2] S. Floyd *et al*, *A Reliable Multicast Framework for Light-weight Sessions and Application Level Framing.* A later version of the ACM SIGCOMM paper (ftp://ee.lbl.gov/papers.srm1.tech.ps.Z), November 1995.

[3] M. Hofmann, *Enabling Group Communication in Global Networks.* Proceedings of Global Networking'97, Calgary, Alberta, Canada, November 1996.

[4] H.W. Holbrook, S.K. Singhal and D.R. Cheriton, *Log-Based Receiver Reliable Multicast for Distributed Interactive Simulation.* Proceedings of ACM SIGCOMM, pages 328-341, August 1995.

[5] S. Kasera, J. Kurose and D. Towsley *Scalable Reliable Multicast Using Multiple Multicast Groups.* Proceedings of ACM Sigmetrics, June 1997.

[6] S. Kasera, J. Kurose and D. Towsley *A Comparison of Server–Based and Receiver–Based Local Recovery Approaches for Scalabale Reliable Multicast.* CMPSCI Tech Report TR 1997-69, December 1997.

[7] B.N. Levine, D.B. Lavo and J. Garcia-Luna-Aceves, *The Case for Reliable Concurrent Multicasting Using Shared Ack Trees.* Proceedings of ACM Multimedia, November 1996.

[8] J.C. Lin and S. Paul, *RMTP: A Reliable Multicast Transport Protocol.* Proceedings of IEEE Infocom 1996.

[9] S. Pingali, D. Towsley and J. Kurose, *A Comparison of Sender-Initiated and Receiver-Initiated Reliable Multicast Protocols.* Proceedings of ACM Sigmetrics, 1994.

[10] D. Tennenhouse, J. Smith, W. Sincoskie, D. Wetherall and G. Minden, *A Survey of Active Network Research.* IEEE Communications Magazine, January 1997.

[11] D. Towsley, J. Kurose and S. Pingali, *A Comparison of Sender-Initiated and Receiver-Initiated Reliable Multicast Protocols.* IEEE Journal on Selected Areas in Communications, 15:398-406, April 1997.

[12] D. Towsley, *An Analysis of a Point-to-Multipoint Channel Using a Go-Back-N Error Control Protocol.* IEEE Transactions on Communications, 33:282-285, March 1985.

[13] M. Yajnik, J. Kurose and D. Towsley, *Packet Loss Correlation in the MBone Multicast Network.* Proceedings of Global Internet Conference, November 1996.

[14] R. Yavatkar, J. Griffioen and M. Sudan, *A Reliable Dissemination Protocol for Interactive Collaborative Applications.* Proceeding of ACM Multimedia, November 1995.