A Tree-Based Reliable Multicast Scheme Exploiting the Temporal Locality of Transmission Errors

Jinsuk Baek[1] Jehan-François Pâris[1]
Department of Computer Science
University of Houston
Houston, TX 77204-3010
{jsbaek, paris}@cs.uh.edu

Abstract

Tree-based reliable multicast protocols provide scalability by distributing error-recovery tasks among several repair nodes. These repair nodes keep in their buffers all packets that are likely to be requested by any of its receiver nodes. We address the issue of deciding how long these packets should be retained and present a buffer management scheme taking into account the fact that most packet losses happen during short error bursts. Under our scheme, receiver nodes do not normally acknowledge correctly received packets and repair nodes routinely discard packets after a reasonable time interval. Whenever a receiver node detects a transmission error, its send a negative acknowledgement to its repair node and start acknowledging up to *k* correctly received packets. Whenever a repair node receives a retransmission request, it stops discarding packets that have not been properly acknowledged until it has received *k* consecutive acknowledgements from each node that had requested a packet retransmission.

I. Introduction

A growing number of network applications require a sender to distribute the same data to a large group of receivers. Multicast is an efficient way to support this kind of applications. One of the most difficult issues in end-to-end multicasting is that of providing an error-free transmission mechanism.

Ensuring reliability requires efficient schemes for retransmission control, flow control, congestion control and so on. This has led to numerous proposals aiming at providing scalable reliable schemes.

Among these proposals, tree-based protocols [1-4, 6-9, 11, 18, 19] are known to provide high scalability as well as reliability. These protocols construct a logical tree at the transport layer. This logical tree comprises three types of nodes: the *sender node*, *repair nodes*, and *receiver nodes*. The sender node is the root of the logical multicast tree. It controls the overall tree construction and is responsible for resending lost packets within the group. Each repair node acts as a local server for a local group of receiver nodes in the tree. It integrates the status information of its receiver nodes and performs local error recovery for these nodes using the data cached in its buffer. As a result, tree-based protocols achieve scalability by distributing the server retransmission workload among the repair nodes.

We believe that the buffers of these repair nodes should be managed in an efficient manner because unnecessary packets stored in their buffer waste storage resources. Schemes addressing this issue can be broadly divided into ACK-based [8, 9, 18, 19] and NAK-based schemes [3, 4, 6].

ACK-based schemes require receiver nodes to send an ACK to their repair node each time they have correctly received a packet. This lets repair nodes discard from their buffers all packets that have been acknowledged by all receiver nodes. ACK-based schemes do not scale well due to the ACK implosion occurring at the repair nodes. Hence, the number of receiver nodes that can be handled by a single repair node will be limited by the repair node ability to handle these ACKs.

NAK-based schemes provide a more scalable solution, because receiver nodes only contact their repair node when they have not correctly received a packet. Unfortunately, these schemes do not provide any efficient mechanism to safely discard packets

[1] Supported in part by the National Science Foundation under grant CCR-9988390.

from the repair node buffers. Hence, the repair node may be unable to resend a packet because the request arrived after the repair node had already discarded the packet from its buffer.

We have recently proposed efficient schemes [1, 2] that eliminate many of these limitations by using both positive and negative acknowledgments to manage these buffers in an efficient manner.

Both schemes assumed that packet losses were independent events that were not correlated with previous transmission failures. As a result, they cannot take into account the temporal locality of packet losses to decide when a repair node can safely discard a given packet.

We propose a more efficient buffer management scheme taking advantage of this temporal locality. It assumes that most transmission errors happen during short error bursts separated by long periods of relatively error-free transmission. Under our scheme, receiver nodes do not normally acknowledge correctly received packets. Whenever a receiver node detects a transmission error, it sends a NAK to their repair node to request the retransmission of that packet. After that, the receiver node will acknowledge all correctly received packets until it has correctly received and acknowledged *k* consecutive packets. Repair nodes normally keep in their buffer recently received packets for a time sufficient to handle a majority of retransmission requests. Whenever they receive a retransmission request, they stop discarding packets whose correct reception has not been acknowledged until they have received *k* consecutive ACKs from each node that had requested a packet retransmission.

Our proposal has two major advantages over previous schemes. First, the amount of feedback from receiver nodes is significantly reduced because each receiver node only sends ACKs when it experiences an error burst. This feature provides scalability, since each repair node will be able to handle more receiver nodes. Second, our proposal guarantees fast recovery of transmission errors, since the packets requested from receiver nodes are almost always available in the buffers of the repair nodes. As a result, the proposed scheme can be broadly applied for various types of applications.

The remainder of this paper is organized as follows. Section II briefly surveys existing reliable multicast protocols. Section III describes our new buffer management scheme. In section IV, we analyze the performance of the proposed scheme. Finally, section V contains our conclusions.

II. Related Work

Retransmission control schemes for reliable multicast protocols essentially differ in the strategies they use for deciding which nodes should buffer packets for retransmission and how long these packets should be retained.

*Scalable Reliable Multicas*t (SRM) [6] is a well-known receiver-initiated multicast protocol that guarantees out-of-order reliable delivery using NAKs from receivers. Whenever a receiver detects a lost packet, it multicasts NAKs to all participants in the multicast session. This allows the nearest receiver to retransmit the packet by multicasting. As a result, the protocol distributes the error recovery load from one sender to all receivers of the multicast session. The sole drawback of the SRM protocol is that all receivers have to keep all packets in their buffer for retransmission.

The first tree-based reliable multicast protocol was the *Reliable Multicast Transport Protocol* (RMTP) [18]. RMTP provides reliable multicast by constructing a physical tree of the network layer. It selects a *designated receiver* (DR) in each local region and makes this receiver responsible for error recovery for all the other receivers in that region. To reduce ACK implosion, each receiver periodically unicasts an ACK to its designated receiver instead of sending an ACK for every received packet. This ACK contains the maximum packet number that each receiver has successfully received. As a result, this periodic feedback policy significantly delays error recovery. Hence, RMTP is not suitable for applications that transmit time-sensitive multimedia data. In addition, RMTP stores the whole multicast session data in the secondary memory of the DR for retransmission, which makes it poorly suited for transfers of large amounts of data. Some of these problems were addressed in RMTP-II [19] by the addition of NAKs.

Guo [8] proposed a stability detection algorithm partitioning receivers into groups and having all receivers in a group participate in error recovery. This is achieved by letting receivers periodically exchange history information about the set of messages they have received. Eventually one receiver in the group becomes aware that all the receivers in the group have successfully received the packet and announces this to all the members in the group. Then all members can safely discard the packet from the buffer. This feature causes high message traffic overhead because the algorithm requires frequent exchange of messages.

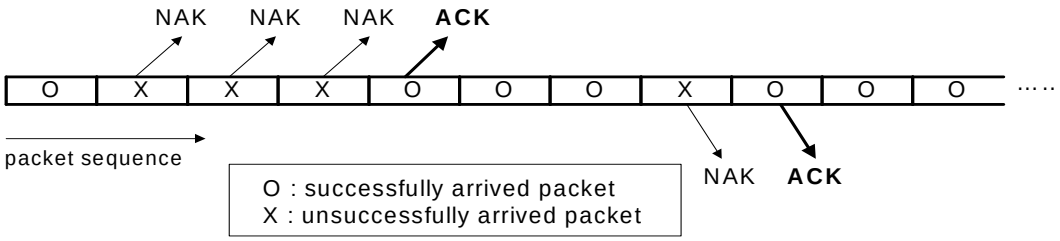

Figure 1. Example of the feedback schedule of a receiver node for *k*=1

The *Randomized Reliable Multicast Protocol* (RRMP) [20] is an extended version of the *Bimodal Multicast Protocol* (BMP) [3]. BMP uses a simple buffer management policy in which each member buffers packets for a fixed amount of time. RRMP uses instead a two-phase buffering policy: feedback-based short-term buffering and randomized long-term buffering. In the first phase, every member that receives a packet buffers it for a short period of time in order to facilitate retransmission of lost packets in its local region. After that, only a small random subset of members in each region continues to buffer the packet. The drawback of this protocol is that it takes a long time for the receiver to search and find the correct repair nodes as the number of participants increase.

The *Search Party* protocol [4] uses a timer to discard the packet from the buffer: each member in the group simply discards packets after a fixed amount of time. The protocol remains vague on the problem of selecting the proper time interval for discarding packets.

Most NAK-based multicast protocols remain equally vague on that issue because the absence of a NAK from a given receiver for a given packet is not a definitive indication that the receiver has received the packet.

We recently proposed a randomized scheme [1] requiring each receiver node to send NAKs to repair nodes to request packet retransmissions. At periodic intervals, they also send randomized ACKs to indicate which packets can be safely discarded from the buffer of their repair node. The scheme reduces delay in error recovery, because the packets requested from the repair nodes are always available in their buffers. In addition, it greatly reduces the number of repair nodes required to handle a given number of receiver nodes. More work is still needed to ascertain the optimal ACK transmission intervals for both receiver nodes and repair nodes.

In our second scheme [2], each repair node discards some packets based on the ACKs from its most unreliable receiver nodes. Like our first scheme, our second scheme does not take advantage of the temporal locality of packet losses.

III. Handling Error Bursts

All previous schemes assume that packet losses are independent events. This is not the case for most real networks. Packet losses tend instead to happen during short error bursts separated by long periods of relatively error-free transmission. There is also a significant spatial correlation in loss among the receiver nodes in a multicast session. In this section, we propose a more efficient buffer management scheme that takes into account the temporal locality of packet losses.

Our scheme assumes a receiver-initiated error recovery process and requires receiver nodes to send a NAK to their repair node every time they detect a packet loss. Thus, a receiver node that does not experience any packet loss will not send back any feedback to its repair node. We refer to this mode of operation as the *normal transmission mode*. Whenever a receiver node detects a transmission error, it sends a NAK to its repair node and switches to a new mode of operation called the *error mode*.

While a receiver node is in error mode, it sends an ACK for each received packet including retransmitted packets. It will stay in that mode until it has correctly received and acknowledged *k* consecutive packets. After that, it will return to the normal transmission mode and cease to acknowledge the packets it receives. Figure 1 illustrates this behavior. In this example, *k* is equal to 1, which means that the receiver node will return to the normal transmission mode once it has received and acknowledged exactly one correct packet.

Repair nodes also have two distinct modes of operation. Under their normal transmission mode,

they keep in their buffer recently received packets for a time sufficient to handle a majority of retransmission requests. Whenever they receive a NAK, they switch to an error mode preventing them from discarding packets that have not been acknowledged by all nodes that have reported a packet loss. They will stay in that mode until they have received k consecutive ACKs from each of this node. After that, they return to their normal transmission mode.

Figures 2 and 3 describe our scheme in more detail. They assume that the multicast tree has N_{RP} repair nodes and that each repair node serves N receiver nodes. Each repair node will maintain:

1. One *error list* containing all the receiver nodes that are currently operating in error mode: the repair node will operate in error mode whenever this list is not empty and in normal mode otherwise;
2. One *ACK list* per acknowledged packet containing all the receiver nodes that have acknowledged a specific packet: these lists only exist when the repair node operates in error mode;
3. One counter per receiver node to keep track of the number of consecutive assignments it should receive from that node before removing it from its error list.

Observe that our scheme assumes that a repair node operating in normal mode will immediately discard any packet that has exceeded its retention time. In practice, we expect these packets to be expelled whenever the repair node schedules a buffer sweep.

Note also that our scheme does not guarantee that every repair node will always have in its buffer all the packets requested by any of its receiver nodes; it only reduces the likelihood of that event. Retransmission failures can still happen when a NAK arrives after the packet it requested was discarded but these failures will only happen when the repair node is in the normal operating mode. This will occur either just at the beginning of an error burst or after the nodes have incorrectly assumed that the current error burst has ended.

The remaining retransmission failures will have to be forwarded to either an upstream repair node or to the sender node itself, depending of the topology of the error-recovery tree. There is little we can do to eliminate retransmission failures happening at the beginning of an error burst. We can, however, eliminate most other retransmission failures by increasing the number k of consecutive ACKs the repair node must receive from a node before removing that node from its error list.

Algorithm:

```
Join multicast group
Set mode to normal
Begin loop
  Switch (event)
   event :  Packet l from sender node arrives
            Store packet in buffer
            Start retention timer for packet l
            Break
   event :  ACK from receiver node i for packet l arrives
            Verify that receiver node i is in error list
            Verify that repair node is in error mode
            Add receiver node i to ACK list of packet l
            If (ACK list of packet l = error list and
                packet is expired)
                Discard packet l
            End if
            Increment counter_i
            If (counter_i = k)
              Remove receiver node i from in error list
            End if
            If (error list is empty)
              Set mode to normal
            End if
            Break
   event :  NAK from receiver node i arrives
            Retransmit missing packet
            Add receiver node i to error list
            Set mode to error
            Reset counter_i
            Break
   event :  Retention timer interrupt for packet l
            If (operating mode is normal)
              Discard packet
            Else
              Mark packet as expired
            End if
            Break
  End switch
End loop
Leave multicast group
```

Figure 2. Algorithm for repair node j ($1 \le j \le N^{RP}$)

Algorithm:

```
Join multicast group
Set mode to normal
  Begin loop
  Switch (event)
    event :  Data from sender node arrives
             Store packet in buffer
             If (mode = error)
               Send ACK to repair node
               Increment counter
               If (counter = k)
                 Set mode to normal
               End if
             End if
             Break
    event :  Missing packet detected
             Send NAK to repair node
             Reset counter
             Set mode to error
             Break
  End switch
End loop
Leave multicast group
```

Figure 3. Algorithm for receiver node i ($1 \le i \le N$)

Receiver nodes that leave the multicast session without giving any notice can disrupt the multicast session for all receiver nodes. The repair node will use a timeout mechanism to detect them and cut them off.

IV. Performance Analysis

In this section, we evaluate the performance of our scheme assuming that all packets in an error burst will always be lost. We assume each receiver has two states, namely, state <1> meaning it has correctly received the last packet and state <0> meaning it has not correctly received that packet. Every time a packet is sent to the receiver, it will experience a transition that could either leave it in its current state or move it to another state. We will focus our discussion to the two transitions leading to state 0, that is <00> and <10>, as they both correspond to a packet loss.

We assume that the probabilities of these transitions follow Easton's model [5] which are given by

$$p_{00} = r + (1 - r)L$$

$$p_{10} = (1 - r)L$$

where r and L are positive integers ≤ 1.

The steady-state probability p_0 of losing a given packet is given by

$$p_0 = p_0 p_{00} + p_1 p_{10}$$

$$= p_0 r + p_0(1 - r)L + (1 - p_0)(1 - r)L,$$

which simplifies into

$$p_0 = p_0 r + (1 - r)L$$

and

$$p_0 = L.$$

Hence, the L parameter represents the steady state probability of not correctly receiving a packet. The r parameter affects the duration of error bursts. With $r = 0$, all packet losses are independent events. When r increases, packet losses become more and more correlated. Let us show how that parameter can be estimated from the average duration of error bursts. The probability that an error burst will affect exactly b packets is then given by

$$P(b \text{ lost packets per error burst}) = 1p_{01} + 2p_{00}p_{01} + 3p_{00}^2 p_{01} + \ldots = \sum_{b=0}^{\infty} (b+1)\, p_{00}^b\, p_{01},$$

which is the mean of a geometric distribution. Hence the mean number of lost packets per error burst is given by

$$\mu = \frac{1}{1 - p_{00}} = \frac{1}{p_{01}} = \frac{1}{1 - r - (1 - r)L},$$

Most networks are fairly reliable and have $L \ll r$. In that case, $p_{00} \approx r$. The equation above can be rewritten as

$$\mu = \frac{1}{1 - r}$$

Hence $r = 0.8$ corresponds to an average number of 5 lost packets per error burst.

IV.1 Feedback Implosion

The first main advantage of our scheme is that it significantly reduces the number of feedbacks sent by receiver nodes to their repair nodes.

Consider now a multicast session involving N receiver nodes $R_1, R_2, \ldots, R_N$ sharing the same repair node T. We assume that these n nodes are subject to independent packet losses with L_i and r_i denoting the respective L and r coefficients of node R_i.

Since all packets in an error burst are always lost, we do not have to consider the possibility that a receiver node may incorrectly assume that the current error burst has ended and can safely select $k = 1$. Each receiver node R_i will thus send to its repair node T:

1. A NAK every time they do not receive a packet; and
2. An ACK for the first packet they receive correctly after having sent one or more NAKs.

Over a session involving the transmission of m packets, the number of feedbacks from a receiver node is given by

$$m(p_0 p_{00} + p_0 p_{01} + p_1 p_{10})$$

The number of feedbacks sent by receiver node R_i to its repair node can be then rewritten as

$$m(L_i + (1 - L_i)(1 - r_i)L_i) \quad (1)$$

Hence the total number F_{BURST} of feedbacks received by the repair node from its N receiver nodes will be given by

$$F_{BURST} = m\sum_{i=1}^{N}(L_i + (1 - L_i)(1 - r_i)L_i) \quad (2)$$

When all link failure probabilities are equal, that is, $L_1 = L_2 = \ldots = L_N = L$, equation (3) simplifies into

$$F_{BURST} = mN\left(L + (1 - L)(1 - r)L\right) \quad (3)$$

Under the same assumptions, the number of feedbacks F_{ACK} for an ACK-based scheme, where all receiver nodes acknowledge all the packets they receive, will be given by

$$F_{ACK} = mN.$$

The difference Δ between the numbers of feedbacks of the two schemes will be given by

$$\Delta = mN(1 - L - (1 - L)(1 - r)L) \quad (4)$$

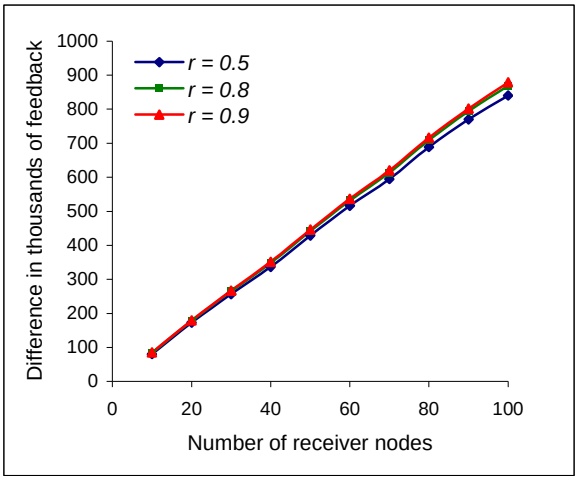

Figure4. Difference Δ vs. the number *N* of receiver nodes per repair node

Figure 4 shows how this difference increases with N for three different values of r when the loss probabilities L_i are uniformly distributed between 0 and 1. We selected the number of transmitted packets m = 10,000, which roughly represents a transfer of 10 megabyte with a packet size equal to 1 kilobyte. When there are 100 receiver nodes, the difference is more than 800,000 feedbacks in all r values. This numerical result indicates that our scheme provides efficient buffer management functionality for the repair node by reducing the number of feedbacks sent by receiver nodes. This feature provides scalability, since each repair node will be able to handle more receiver nodes.

IV.2 Additional Retransmissions

Whenever the repair node T receives a NAK from receiver node R_i, it will switch to error mode and cease discarding packets until they have been acknowledged by all receiver nodes that operate in error mode. We assume the failure of receiver nodes and their repair node is not correlated.

An upper bound for the probability $P(M_{BURST})$ that T will not have in its buffer a packet that is requested by a receiver node R_i is then given by the probability that receiver node R_i enters an error burst or the repair node did not correctly receive the requested packet.

$$P(M_{BURST}) = p_1 p_{10} + p_0 p_{00} L_{rp}$$
$$= (1 - L_i)(1 - r_i)L_i + L_i(r_i + (1 - r_i) L_i) L_{rp} .$$

where L_{rp} is the packet loss probability of the repair node.

This upper-bound is extremely pessimistic because it assumes that the repair node will never be able to find in buffer the first packet of any error burst. This is not true because repair node T will always keep in its buffer all the packets it receives for a reasonable time interval. Hence, the requested packets are available at the repair node if the NAKs arrive before the timer is expired. Let us call this probability A. The probability might be very close to 1 if the repair node has large enough timer value. If we assume that A is equal to 0.9, the repair node will only be unable to deal with 10% of the retransmission requests sent by other nodes, because the requested packet will be removed before any NAK arrives.

In addition, the packet could still be in T's buffer because T was waiting for the ACK of another receiver node that was already inside an error burst. Hence, a more realistic estimate of the probability $P(M_{BURST})$ for N receiver node is given by

$$\begin{aligned} P(M_{BURST}) = {} & p_1 p_{10} \times [\, P(\text{NAK was lost}) \\ & + P(\text{NAK was not lost but repair node did not correctly receive the packet}) \\ & + P(\text{NAK was not lost and repair node correctly receive the packet but NAK did not arrive on time and no other receiver node was in error mode})] \\ & + p_0 p_{00} L_{rp} \\ = {} & p_1 p_{10}[L_i + (1 - L_i) L_{rp} \\ & + (1 - L_i)(1 - L_{rp})(1 - A) \prod_{\substack{j=1 \\ j \neq i}}^{N} (1 - L_j) \,] \\ & + p_0 p_{00} L_{rp} \text{ , for} \leq i, j \leq N \\ = {} & (1 - L_i)(1 - r_i) L_i \,[L_i + (1 - L_i) L_{rp} \\ & + (1 - L_i)(1 - L_{rp})(1 - A) \prod_{\substack{j=1 \\ j \neq i}}^{N} (1 - L_j) \,] \\ & + L_i(r_i + (1 - r_i) L_i) L_{rp} \end{aligned} \quad (5)$$

In NAK-based schemes using a timer mechanism, repair nodes discard packets from their buffers after a time interval. Under same assumptions, the packet missing probability $P(M_{NAK})$ for NAK-based scheme can be given by

$$\begin{aligned} P(M_{NAK}) = {} & L_i \times [\, P(\text{NAK was lost}) \\ & + P(\text{NAK was not lost but repair node did not correctly receive the packet}) \\ & + P(\text{NAK was not lost and repair node correctly receive the packet but NAK did not arrive on time})] \\ = {} & L_i \,[L_i + (1 - L_i) L_{rp} + (1 - L_i)(1 - L_{rp})(1 - A)] \end{aligned} \quad (6)$$

Given the difficulty of finding a closed-form expression for the parameter A, we decided to simulate the behavior of a system with 100 receiver nodes per repair node. To generate the loss probability of each receiver node, we applied the formula $S = 1.22/(\ RTT_{s,i}\sqrt{L_i}\ )$ (from [13]), where S is the packet sending rate in packets/sec, $RTT_{s,i}$ is the round trip time from the sender node to receiver node R_i and L_i is the loss probability between the sender node and receiver node R_i. This assumes that the sender node transmits packets in a TCP-friendly manner and each node in the multicast session uses the UDP protocol.

We simulated the round-trip times $RTT_{s,i}$ as Poisson random variables, each having mean *Avg_RTT*. Similarly, the one-way transit times $OTT_{i,T}$ between a receiver node R_i and its repair node T were also simulated by Poisson random variables with mean *Avg_OTT*. We generated the packet loss probability for each receiver node when the packet-sending rate S is 128 packets per second, average round trip time *Avg_RTT* is 40 ms and average one-way transit time *Avg_OTT* is 15 ms. Figure 5 shows our measurement of the packet loss probability for 100 receiver nodes.

In more actual networks, the underlying transport protocol needs to detect packet duplications especially in case of retransmission. Hence, a dynamic estimation algorithm for NAK timer value should be provided for effective detection of feedbacks. Since we are only interested in the availability of the packet at the repair node, we assumed in our simulation that each receiver node sets its NAK_TIMER value to 40ms, which is an average value of the current *RTT* values.

Using these configuration parameters, we can evaluate the probability that a requested packet will not be present in the repair node. Figure 6 shows how the number of receiving nodes per repair node affects the probability of not finding a requested packet in the repair node buffer. We can see that the NAK-based scheme performs significantly worse than our scheme. We also can see that our scheme always achieves very low packet missing probabilities for all number of receiver nodes per repair node. The probability is below 10^{-4} when there are 100 receiver nodes. This result means the repair node will send only single NAK to its upstream repair node when the sender node transmits 10 megabytes data. In addition, our simulations also indicate that the lowest packet missing probabilities are achieved whenever there are at least 40 receiver nodes per repair node.

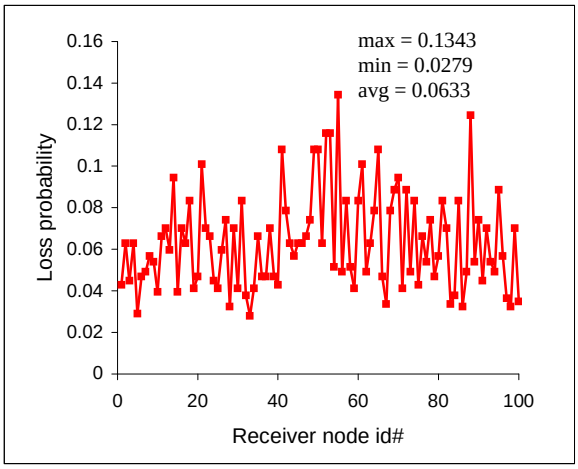

Figure 5. Simulated loss probability

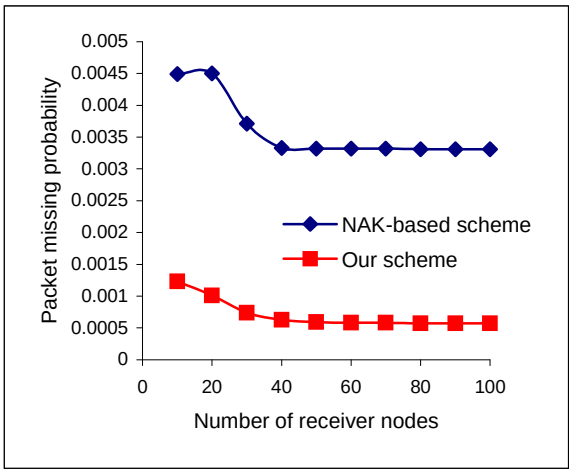

Figure 6. Packet missing probability

Additional retransmissions increase the error recovery delay, since the repair node cannot retransmit the requested packet immediately. The packets will then have to be retransmitted by either the original sender node or upstream repair node. As a result, the error recovery delay could be doubled or even tripled. Also, these additional retransmissions cause unnecessary traffic between the repair nodes.

The performance of NAK-based scheme will improve whenever the repair nodes have very large buffers as well as a long enough timer values. However, this would result in an inefficient use of the available buffer space, because too many packets will remain in buffer for a long time. In addition, the absence of an efficient buffer management scheme is likely to cause sooner or later buffer overflow.

V. Conclusions

We have presented a new reliable tree-based multicast scheme that takes into account the temporal locality of packet losses to limit both the number of feedbacks sent by receiver nodes to their repair nodes and the probability that a given repair node will not be able to handle a given packet retransmission request.

Our scheme operates in two possible modes. In normal mode, receiver nodes do not acknowledge correctly received packets and repair nodes routinely discard packets after a reasonable time interval. Whenever a receiver node detects a lost packet, it sends a NAK to its repair node and switches to error mode. It will then acknowledge all incoming packets and keep operating in that mode until it has correctly received and acknowledged *k* consecutive packets. Similarly, any repair node that receives a NAK from any of its receiver nodes will start operating in error mode and stop discarding packets that have not been properly acknowledged by all receiver nodes that operate in error mode.

Our scheme requires fewer feedbacks than ACK-based schemes because receiver nodes only send ACKs to their repair node when they are in the middle of an error burst. At the same time, it allows the repair nodes to handle more retransmission requests than NAK-based schemes. As a result, our scheme provides an attractive compromise between ACK-based and NAK-based schemes.

References

[1] J. Baek, J. F. Pâris, "A Buffer Management Scheme for Tree-Based Reliable Multicast Using Infrequent Acknowledgments," *Proc. 23rd IEEE International Performance Computing and Communications Conference*, pp. 13-20, April 2004.

[2] J. Baek, J. F. Pâris, "A Heuristic Buffer Management Scheme for Tree-Based Reliable Multicast," *Proc. 9th IEEE Symposium on Computers and Communications*, pp. 1123-1128, June-July 2004.

[3] K. P. Birman *et al.*, "Bimodal Multicast," *ACM Transactions on Computer Systems*, 17(2):41–88, May 1999.

[4] M. Costello and S. McCanne, "Search Party: Using Randomcast for Reliable Multicast with Local Recovery," *Proc. 18th IEEE Conference on Computer Communications*, pp. 1256–1264, March 1999.

[5] M. C. Easton, "Model for database reference Strings Based on Behavior of Reference Clusters," *IBM Journal Research and Development*, 22(2):197–202, March 1978.

[6] S. Floyd *et al.*, "A Reliable Multicast Framework for Lightweight Sessions and Application-Level Framing," *IEEE/ACM Transactions on Networking*, 5(6):784–803, December 1997.

[7] T. Gemmel *et al.*, "The use of Forward Error Correction in Reliable Multicast," IETF draft-ietf-rmt-info-fec-02.txt, October 2002.

[8] K. Guo, and I. Rhee, "Message Stability Detection for Reliable Multicast," *Proc. 19th IEEE Conference on Computer Communications*, pp. 814–823, March 2000.

[9] M. Kadansky *et al.*, "Reliable Multicast Transport Building Block: Tree Auto-Configuration," IETF Internet Draft, draft-ietf-rmt-bb-tree-config-01.txt, November 2000.

[10] S. K. Kasera, J. Kurose, and D. Towsley, "Buffer Requirements and Replacement Polices for Multicast Repair Service," *Proc. 2nd Network Group Communication Workshop* (NGC 2000), pp. 5-14, Nov. 2000.

[11] S. J. Koh *et al.*, "Configuration of ACK Trees for Multicast Transport Protocols," *ETRI Journal*, 23(3):111–120, September 2001.

[12] B. Levine, and J. J. Garcia-Luna-Aceves, "A Comparison of Reliable Multicast Protocols," *ACM Multimedia Systems Journal*, 6(5): 334–344, August 1998.

[13] J. Mahdavi and S.Floyd, "TCP-friendly unicast rate-based flow control," Jan. 1997. http://www.psc.edu/networking/papers/tcp_friendly.html

[14] C. Maihofer and K. Rothermel, "A Robust and Efficient Mechanism for Constructing Multicast Acknowledgment Trees," *Proc. 8th IEEE International Conference on Computer Communications and Network*s, pp. 139–145, October 1999.

[15] M. Mathis, J. Mahdavi, S. Floyd, and A. Romanow, "TCP Selective Acknowledgement Options," RFC 2018, October 1996.

[16] S. Pingali, D. Towsley, and J. F. Kurose, "A Comparison of Sender-Initiated and Receiver-Initiated Reliable Multicast Protocols," *IEEE Journal on Selected Areas in Communications*, pp. 221–230, April 1997.

[17] R. van Renesse, Y. Minsky, and M. Hayden, "A Gossip-Style Failure Detection Service," *Proc. MIDDLEWARE '98*, pp. 55–70, Sept. 1998.

[18] J. C. Lin and S. Paul, "RMPT: A Reliable Multicast Transport Protocol," *Proc. 15th IEEE Conference on Computer Communications* (INFOCOM 96), pp. 1414-1424, March 1996.

[19] B. Whetten and G. Taskale, "The Overview of Reliable Multicast Transport Protocol II," *IEEE Networks*, 14(1):37–47, Jan.-Feb. 2000.

[20] Z. Xiao, K. P. Birman, R. Renesse, "Optimizing Buffer Management for Reliable Multicast," *Proc. 2002 International Conference on Dependable Systems and Networks*, pp. 187–202, June 2002.

[21] R. Yavatkar, J. Griffieon, M. Sudan, "A Reliable Dissemination Protocol for Interactive Collaborative Applications," *Proc. 3rd ACM International Conference on Multimedia*, pp. 333–344, November 1995.