A Retransmission Control Scheme for Tree-Based Reliable Multicast

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

Keywords: Reliable multicast, Tree-based protocol, Missing Probability, Additional Retransmission

Abstract

Tree-based reliable multicast protocols provide scalability by distributing error-recovery tasks among several repair nodes. These repair nodes integrate the status information of their receiver nodes and perform local error recovery for these nodes using the data stored in their buffers. NAK-based error control schemes provide scalable solution by shifting error detection tasks from the repair node to each receiver node. However, they provide no efficient mechanism to safely discard packets from the repair node buffers. This leaves the repair nodes with a difficult choice. They can err on the safety side and keep in their buffer packets that have already been correctly received by all nodes. Conversely, they can discard packets without knowing if they are still needed. Receiver nodes must then obtain these packets from their upper-stream repair node, which could result in a NAK implosion at these upper-stream repair nodes and a much slower repair process. We propose an efficient retransmission control scheme without all the disadvantages of previous schemes. Under our scheme, most of additional retransmissions are performed within a local group. This feature satisfies the original goal of tree-based protocol. Simulation results indicate our scheme significantly reduces NAK implosion and provides fast recovery of transmission error.

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.

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

Among these proposals, tree-based protocols 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: a 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 subset 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. Hence, tree-based protocols achieve scalability by distributing the server retransmission workload among the repair nodes.

Since there might be a large number of repair nodes, their buffers should be managed in an efficient manner. Schemes addressing this issue can be broadly divided into ACK-based [5, 6, 15, 16] and NAK-based schemes [1, 2, 3].

In ACK-based schemes, the receiver nodes send an ACK to their repair node every time they have correctly received packet. This allows each repair node to discard from its buffer all packets that have been acknowledged by all receiver nodes. However, this ACK-based approach does not scale well due to the ACK implosion occurring at the repair nodes. Hence, the repair node's ability to handle these ACKs limits the number of receiver nodes participating in a multicast session.

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

Most tree-based protocols require these missing packets to be retransmitted by some upper-stream repair nodes. Unfortunately, these additional retransmissions can lead to NAK implosion at the upper-stream repair nodes. In addition, they increase the error recovery delay for each receiver node that requests retransmission for the damaged or lost packet, because the repair node cannot retransmit the requested packet immediately. If the upper-stream repair node does not have the packets in its buffer, the requests

will reach the original sender node. In this case, the error recovery delay is unacceptable for time-sensitive applications. Also, these additional retransmissions will increase the sender's workload.

We propose an efficient scheme to control these additional retransmissions in NAK-based schemes. Under our scheme, each repair node predicts which receiver nodes are likely to have the missing packets. The missing packets are retransmitted by these receiver nodes rather than by the original sender node or some upper-stream repair node.

Our proposal has two major advantages over other schemes. First, it keeps most retransmission requests within the local group. This feature satisfies the original goal of tree-based protocols, because each local group performs the error recovery by itself. In addition, it reduces NAK implosion at the upper-stream repair nodes. Second, it provides fast recovery of transmission errors, since most of the packets requested by receiver nodes are retransmitted from one of the members of its local group. 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 retransmission control scheme. In section IV, we show the performance of the proposed scheme. Finally, section V contains our conclusions.

II. RELATED WORK

This section describes the retransmission control schemes of existing reliable multicast protocols. These 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) [3] 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) [15]. RMTP provides reliable multicast by constructing a physical tree of the network layer. It allocates 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. Unfortunately, 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 [16] by the addition of NAKs.

Guo [5] 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 buffer. This feature causes high message traffic overhead because the algorithm requires frequent exchange of messages.

The *Randomized Reliable Multicast Protocol* (RRMP) [17] is an extended version of the *Bimodal Multicast Protocol* (BMP) [1]. 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 locate the correct repair nodes when the number of participants increases.

Finally, the *Search Party* protocol [2] 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.

III. RETRANSMISSION CONTROL

In this section, we show that additional retransmissions will occur when the repair node manages its buffer using a NAK-based scheme, and we describe how our retransmission control scheme avoids all the disadvantages caused by the additional retransmissions from the sender node or upper-stream repair nodes.

III.1. Additional Retransmission Issues

In NAK-based schemes, the repair node batches NAKs for a packet and retransmits the packet periodically as long as there is a pending NAK for that packet. Hence, they must require the repair nodes to buffer all packets for an infinitely long amount of time to achieve full coverage of all retransmission requests by the repair node. As a result, the additional retransmissions cannot be avoided as long as the buffer size is finite.

In NAK-based schemes using a timer mechanism, repair nodes discard some packets from its buffer after a time interval I without considering whether these packets were received by all its receiver nodes. As a result, some packets might be removed from the repair node buffer while their retransmission could still be requested by some of its receiver nodes.

One possible solution would be to let the repair node request the packet from its upper-stream repair node. Unfortunately, this solution cannot guarantee that this upper-stream repair node has the packet. Consider for instance the case where all repair nodes specify the same timer value and discard the same packets at the same time. Also, we need to consider that the upper-stream repair node serves receiver nodes that have more reliable and faster connections. That repair node would be tempted to remove the packets faster than the current repair node by adjusting the timer value according to the network condition. As a result, most of the requested packets will have to be resent by the sender node. This results in unnecessary transmissions, decreasing the whole network performance.

III.2. Our Retransmission Control Scheme

We propose a heuristic approach that provides an efficient way to control retransmissions. Our objective is to reduce the number of additional retransmissions sent by upper-stream repair nodes serving other group of receiver nodes. Hence, our heuristic scheme will allow most retransmissions to be handled within the local group. We make the following assumptions:

- There are N receiver nodes for one repair node. Hence, the repair node is responsible for resending the packets requested with NAKs from N receiver nodes.
- Each of the N receiver node attached to a repair node has an independent packet loss probability L_i for $i = 1, 2,\ldots, N$.
- Each of these receiver nodes has an independent NAK timer NAK_TIMER_i.
- The repair node also has an independent packet loss probability L.
- The repair node has an independent NAK timer as well as a PACKET_DISCARD timer it uses to decide when to discard packets from its buffer.

Our basic idea is that every group of receivers will have one or more members that experience lower error probability than the other members of the group. Hence any packet that is requested by other members but has already been discarded by the repair node will be likely to be correctly received by one of these members. To achieve this, each repair node collects status information from its N receiver nodes and selects R *representative nodes* as follows: each repair node selects R most reliable receiver nodes among the N receiver nodes it serves. These R nodes will be the R receiver nodes with the lowest packet loss probabilities L_i for $i = 1, 2,\ldots, N$. $\mathcal{R}$ is the set containing these R most reliable receiver nodes and $|\mathcal{R}| = R$.

These selections are not static. The repair node will periodically reselect new representative nodes to adapt to dynamic network events, such as, aborted connections of one or more representative nodes, dynamic joins and leaves.

These representative nodes will be asked to buffer some received packets for a finite interval of time. This will ensure that the packets, whose retransmission are requested by any other receiver nodes after they have been discarded by the repair node, will be almost always available in the buffer of one of these representative nodes.

The repair node requests the packets to the representative node that has the lowest loss probability. Then, the representative node responds to the repair node with that packet if it has successfully received the packet. The repair node can now retransmit the packet to the receiver nodes, which requested the packet.

Let us turn our attention to the probability a repair node is not able to retransmit a requested packet. Let be M_{NAK} the probability that the repair node does not have the requested packet in its buffer.

There are two cases to consider. First, the requested packet can be missing because the repair node never received it. Hence, additional retransmissions will be required until the packet arrives at the repair node. We call this case M_1. If we assume all receiver nodes have a feedback loss probability equal to their packet loss probability L_i, the probability $P(M_1)$ can be given by

$$
\begin{aligned}
P(M_1) &= P(\text{the repair node did not receive the packet}) \\
&\quad \times P(\text{some other nodes did not receive the packet and their NAKs were not lost}) \\
&= P(\text{the repair node did not receive the packet}) \\
&\quad \times (1 - P(\text{all receiver nodes either received the packet or did not receive the packet and their NAKs were lost})) \\
&= L\left(1 - \prod_{i=1}^{N} (1 - L_i + L_i^{\,2})\right) \qquad (1)
\end{aligned}
$$

Second, the packet will not be available if the receiver nodes request it after the repair node has already discarded it. We call this case M_2. However, some packets could still be available if other NAKs for the same packet 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 a 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. We also need to take into account the impact of lost NAKs. If the NAKs of all the receiver nodes that did not receive the packet fail to reach the repair node, then the repair node will discard the packet before it receives a second request for that packet from one of the receiver nodes. This probability $P(M_2)$ can be given by

$$\begin{aligned} P(M_2) &= P(\text{the repair node correctly received the packet}) \\ &\quad \times (1-A) \times P(\text{some receiver nodes did not receive the packet}) \\ &\quad + P(\text{the repair node correctly received the packet}) \\ &\quad \times A \times P(\text{all NAKs sent by the receiver nodes that did not receive the packet were lost}) \\ &= (1-L)(1-A)(1-\prod_{i=1}^{N}(1-L_i)) \\ &\quad + (1-L)A(\prod_{i=1}^{N}(1-L_i+L_i^2) - \prod_{i=1}^{N}(1-L_i)) \end{aligned} \quad (2)$$

Hence, a realistic estimate of the packet missing probability $P(M_{NAK})$ is given by

$$P(M_{NAK}) = P(M_1) + P(M_2) \quad (3)$$

Observe that in the first case (M_1), the repair node has not received the packet that is requested by one or more of its receiver nodes. In that case, the repair node will send a single NAK to its upper-stream repair node and hold on the NAKs for its receiver nodes as it knows it will be able to service them soon enough. The requested packets from receiver nodes will be retransmitted from the selected representative nodes. The single NAK from the repair node will not be counted as an additional retransmission, because the upper-stream repair node makes no distinction between repair node and receiver nodes. Therefore, no additional request goes outside from the local group. Also, this feature eliminates NAK implosion problem at the upper-stream repair node.

For the second case, we will use the representative nodes, since these nodes have the lowest loss probability. Hence, the packets, requested but not available in repair node's buffer, will be retransmitted from one of these representative nodes. If we assume all representative nodes have a long enough timer value for discarding packets, the packet missing probability of our heuristic scheme can be given by

$$\begin{aligned} P(M_{Heuristic}) &= P(M_{NAK}) \\ &\quad \times P(\text{no representative node received the packet}) \\ &= P(M_{NAK}) \times \prod_{i \in \mathcal{R}} L_i \end{aligned} \quad (4)$$

Assumption: The repair node removes some packets from its buffer after a specified time interval.

Algorithm:

```
Join multicast group
Calculate L_i for all its receiver nodes, 1 ≤ i ≤ N
Set representative nodes and send REP message
Begin loop
 Switch (event)
      event :  Packet from sender node arrives
               Store packet in buffer
                 If (packet is requested by receiver nodes)
                     retransmit it
               Endif
               Break
      event :  NAK from a receiver node arrives
               If (packet is available) retransmit it
               Else get packet from a representative node
               Endif
               Break
      event :  Packet from representative node arrives
               Store packet in buffer and retransmit it
               Break
      event :  Packet from representative node has not arrived
               within a specified time interval
               Request it from sender node
               Break
 End switch
End loop
Leave multicast group
```

Figure 1. Repair node algorithm

Algorithm:

```
Join multicast group
Set count = 0
Begin loop
  Switch (event)
      event :  Packet from sender node arrives
               Store packet in buffer
               Break
      event :  Missing or damaged packet is detected
               Send NAK to repair node
               Break
      event :  Control message from repair node arrives
               If (REP) then mode = REP else mode = N_REP
               Endif
               Break
      event :  NAK from a repair node arrives
               If (available and mode = REP) Transmit packet
               Endif
               Break
  End switch
End loop
Leave multicast group
```

Figure 2. Receiver node algorithm

Even when A=1 and the repair node has representative nodes whose loss probability is close to 1, we need to consider the case when the repair node receives NAKs for a packet that has already been discarded from its buffer and no representative node has correctly received the packet. As described in the previous subsection, these packets might not be available in its upper-stream repair node. Therefore, our scheme requires the repair node to request all missing packets directly from the original sender node. The workload of that node will remain reasonable as long as $P(M_{Heuristic})$ is kept low enough.

Figures 1 and 2 describe the respective algorithms for repair and receiver nodes.

IV. PERFORMANCE ANALYSIS

In this section, we show the performance of the proposed retransmission control scheme by computer simulation. All the simulation experiments are performed for up to 100 receiver nodes per repair node.

IV.1. Additional Retransmissions

The proposed scheme requires the repair node to request the missing packets to some representative nodes to be selected on the base of their connection reliability. Hence, it does not require many additional retransmissions either from its upper-stream repair nodes or sender node, because the repair node will have in its buffer most packets that can be requested by any of its receiver nodes. This feature provides fast error recovery for receiver nodes and reduces network traffics between the repair nodes.

We evaluate how many additional retransmissions are required in NAK-based scheme using timer mechanism for discarding packets. The number of additional retransmissions is depending on the packet missing probability $P(M_{NAK})$, which will vary between

$$P(M_{NAK}) = L(1-\prod_{i=1}^{N}(1-L_i+L_i^{\,2}))+(1-L)(1-\prod_{i=1}^{N}(1-L_i))$$

for $A = 0$, and

$$P(M_{NAK}) = L(1-\prod_{i=1}^{N}(1-L_i+L_i^{\,2})) + (1-L)(\prod_{i=1}^{N}(1-L_i+L_i^{2})-\prod_{i=1}^{N}(1-L_i))$$

for $A = 1$

Under the same assumptions, the packet missing probability $M_{Heuristic}$ for the proposed scheme is given by

$$P(M_{Heuristic}) = P(M_{NAK})\prod_{i\in R} L_i \qquad (5)$$

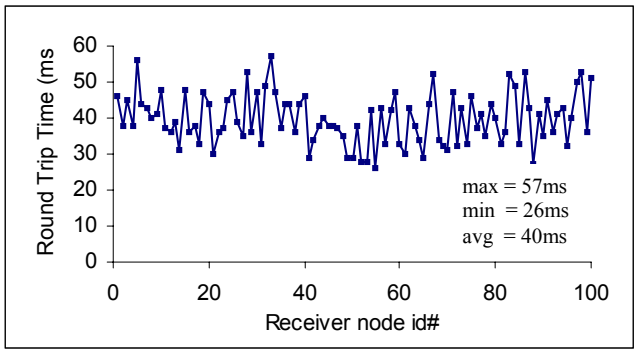

Figure 3. Simulated round trip time

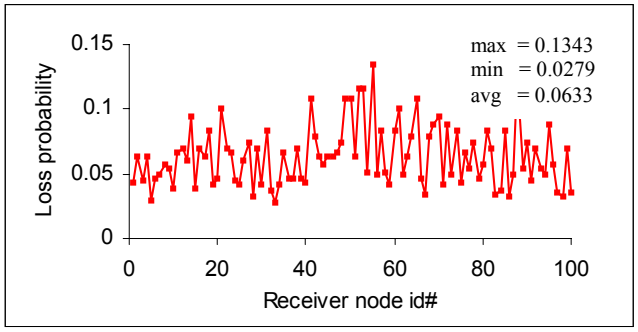

Figure 4. Simulated loss probability

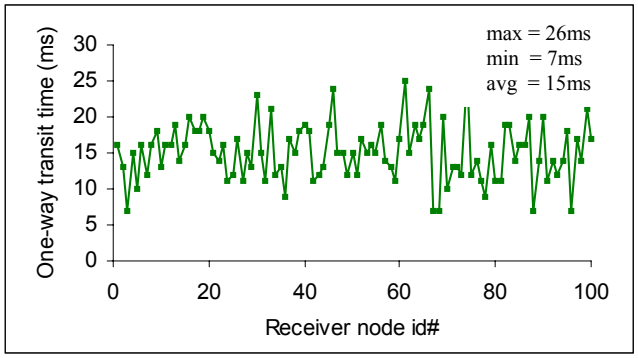

Figure 5. Simulated one-way transit time

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. The parameters of this model are summarized in table I.

To generate the loss probability of each receiver node, we assumed that the sender node transmitted the packets in TCP-friendly manner and applied the formula $S = 1.22/(RTT_{s,i}\sqrt{L_i})$ (from [10]), 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 i and L_i is the loss probability between the sender node and receiver node i.

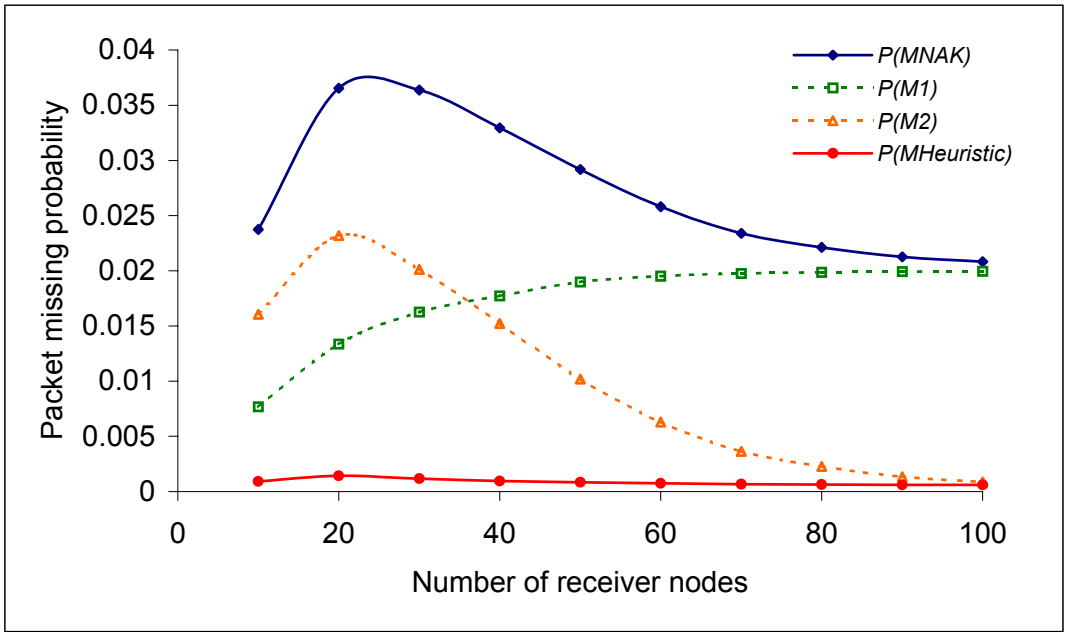

Figure 6. Packet missing probability

We simulated the round-trip times $RTT_{s,i}$ as Poisson random variables each having mean *Avg_RTT*. Similarly, the one-way transit times $OTT_{i,rp}$ between a receiver node *i* and its repair node *rp* were also modeled by Poisson random variables with mean *Avg_OTT*. Figure 3, 4, and 5 respectively show our measurements for roundtrip time, loss probability and one-way transit time for 100 receiver nodes.

Using the configuration parameters in Table I, we evaluated the probability that a requested packet will not be present in the repair node. In particular, we compared the performance of our scheme with that of a NAK-based scheme keeping all packets in the repair node for 60 ms then in the representative nodes buffer for 40 additional ms. Recall that the repair node should select the receiver node with the most reliable connections as representative node. Since our model assumed that all nodes had identical packet loss probabilities, the representative node was identical in all respects to the other receiver nodes.

We are thus evaluating our scheme under very unfavorable conditions as we do not endow the representative node with any special properties. In most real situations, some receiver nodes will experience more reliable links than the other nodes. Selecting these "better" receiver nodes as representatives will provide even lower missing packet probabilities.

Figure 6 shows how the probability of not finding a requested packet in the repair node buffer is affected by the number of receiver nodes per repair node. We can see that

Sending rate *S*	128packets/second
Avg_RTT	40ms
Avg_OTT	15ms
N	100 receiver nodes
R	3 representative nodes
L	0.02
NAK_TIMER_i, $1 \le i \le N$	30ms
M	10,000 packets (≅10Mbyte)

Table I. Configuration Parameters

the performance of the NAK-based scheme is getting better as the number of receiver node increases. However, we should also observe that the missing probability is not significantly reduced, because the $P(M_1)$ consistently increases with the group size.

We also see that our heuristic scheme works well by showing the missing probability becomes progressively close to 0 percent when the number of receiver nodes per repair node increases above 20. It is about 10^{-5} when there are 100 receiver nodes including one representative node and two backup representative nodes.

To provide the same performance as our scheme, other NAK-based schemes would require the repair nodes to have very large buffers as well as a long enough timer values.

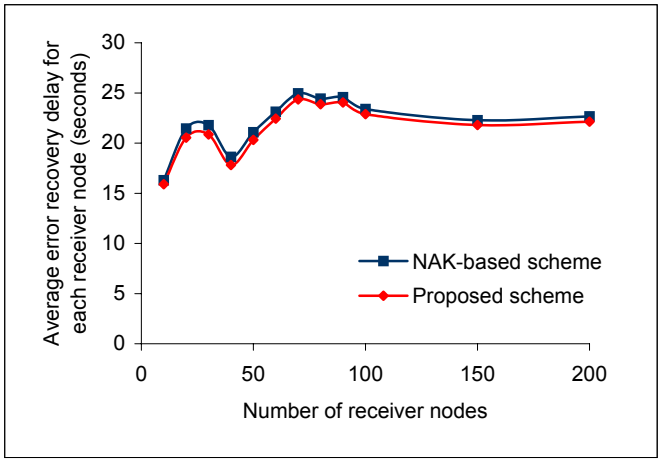

Figure 7. Difference between the error recovery delays of our scheme and a NAK-based scheme

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.

IV.2. Error Recovery Delay

Packet retransmissions involving upstream nodes also increase the error recovery delay, because the packets must then be retransmitted from either the original sender node or another upstream repair node. As a result, the error recovery delay could be doubled or even tripled. Also, these additional retransmissions cause unnecessary transmissions higher up in the recovery tree.

This effect becomes more important when the height of the repair tree increases. In order to reach a reasonable evaluation of the error recovery delays of the two schemes, we assumed that the height of the repair tree was equal to 3. That is, the tree comprises a sender node, receiver nodes, repair nodes, and upper-stream repair nodes. We also assume that the round trip time between each type of nodes is 30 ms. In order to evaluate the minimum delay difference between both schemes, we assume that the additional retransmission for NAK-based scheme is always correctly transmitted from the upper-stream repair node. Otherwise, the difference would be even more important.

We measured the average recovery delay experienced by the receiver nodes under these assumptions. These results are shown in Figure 7. Even under simulated parameter values, the proposed scheme performs slightly better than the NAK-based scheme by showing the delay difference varying from 395 ms to 928 ms. Our data also indicate that the most reasonable number of receiver nodes per repair node is equal to 40.

Note that the difference decreases as the number of receiver nodes per repair node increases as the packet missing probability of the NAK-based scheme decreases. However, the difference increases when the repair node has more than 150 receiver nodes because the $P(M_1)$ consistently increases with the group size.

V. CONCLUSION

We have proposed a heuristic retransmission control scheme for NAK-based buffer management scheme that provides scalability and reliability in a multicast session. As in all NAK-based schemes, receiver nodes only contact their repair node to request packet retransmissions. Our scheme increases the probability that these retransmission requests will be satisfied locally by associating with each repair node a few *representative nodes* among the nodes with the most reliable links. These representative nodes will be asked to keep received packets on hand for up to twice the time repair node retention time. Whenever a repair node does not have in its buffer a packet that it has to retransmit, it will request the missing packet from one of these representative nodes before contacting an upstream repair node. Our scheme greatly reduces the number of additional retransmissions outside the local group, because the packets requested from the receiver nodes are almost always available in their buffers. As a result, it provides fast error recovery and eliminates unnecessary transmissions higher up in the recovery tree.

REFERENCES

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

[2] M. Costello and S. McCanne, "Search Party: Using Randomcast for Reliable Multicast with Local Recovery," *Proceedings of the 18th IEEE Conference on Computer Communications* (INFOCOM '99), pp. 1256–1264, New York, USA, March 1999.

[3] 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.

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

[5] K. Guo, and I. Rhee, "Message Stability Detection for Reliable Multicast," *Proceedings of the 19th IEEE Conference on Computer Communications* (INFOCOM 2000), pp. 814–823, New York, USA, March 2000.

[6] 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.

[7] S. K. Kasera, J. Kurose, and D. Towsley, "Buffer Requirements and Replacement Polices for Multicast Repair Service", *Proceedings of the 2nd Network Group Communication Workshop* (NGC 2000), pp. 5-14, Stanford University, USA, November 2000.

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

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

[10] J. Mahdavi and S.Floyd, "TCP-Friendly Unicast Rate-Based Flow Control," January 1997. http://www.psc.edu/networking/papers/tcp_friendly.html

[11] C. Maihofer and K. Rothermel, "A Robust and Efficient Mechanism for Constructing Multicast Acknowledgment Trees," *Proceedings of the 8th IEEE International Conference on Computer Communications and Network*s, pp. 139–145, Boston-Natick, USA, October 1999.

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

[13] 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.

[14] R. Renesse, Y. Minsky, and M. Hayden, "A Gossip-Style Failure Detection Service," *Proceedings of MIDDLEWARE '98*, pp. 55–70, The Lake District, England, September 1998.

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

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

[17] Z. Xiao, K. P. Birman, R. Renesse, "Optimizing Buffer Management for Reliable Multicast," *Proceedings of the International Conference on Dependable Systems and Networks* (DSN'02), pp. 187–202, Washington, D.C., USA, June 2002.

[18] R. Yavatkar, J. Griffioen, M. Sudan, "A Reliable Dissemination Protocol for Interactive Collaborative Applications," *Proceedings of the 3rd ACM International Conference on Multimedia*, pp. 333–344, San Francisco, USA, November 1995.

Jinsuk Baek obtained his B.S. and M.S. degrees in Computer Science from Hankuk University of Foreign Studies (HUFS) in Yongin, Korea. He is currently a Ph.D. candidate in the Department of Computer Science of the University of Houston in Houston, Texas. His research interests include scalable reliable multicast protocols, congestion control algorithms proxy caching systems and mobile computing.

Jehan-François Pâris obtained his M.S. in Chemical Engineering from the Université Libre de Bruxelles in Brussels, Belgium and Ph.D. in Computer Science from the University of California, Berkeley. He is currently professor of computer science at the University of Houston in Houston, Texas. His research interests include memory hierarchies, scalable reliable multicast protocols, distribution protocols for video-on-demand, and distributed systems security.