A Heuristic Buffer Management and Retransmission Control Scheme for Tree-Based Reliable Multicast

Jinsuk Baek and Jehan-François Pâris

We propose a heuristic buffer management scheme that uses both positive and negative acknowledgments to provide scalability and reliability. Under our scheme, most receiver nodes only send negative acknowledgments to their repair nodes to request packet retransmissions while some representative nodes also send positive acknowledgments to indicate which packets can be discarded from the repair node's buffer. Our scheme provides scalability because it significantly reduces the number of feedbacks sent by the receiver nodes. In addition, it provides fast recovery of transmission errors since the packets requested from the receiver nodes are almost always available in their buffers. Our scheme also reduces the number of additional retransmissions from the original sender node or upstream repair nodes. These features satisfy the original goal of tree-based protocols since most packet retransmissions are performed within a local group.

Keywords: Reliable multicast, tree-based protocol, missing probability, additional retransmission, recovery delay.

Manuscript received Apr. 5, 2004; revised July 29, 2004.

This work was supported in part by the National Science Foundation under grant CCR-9988390.

Jinsuk Baek (phone: +1 281 752 5796, email: jsbaek@cs.uh.edu) and Jehan-François Pâris (email: paris@cs.uh.edu) are with the Computer Science Department, University of Houston, USA.

I. Introduction

A growing number of network applications require a sender to distribute the same data to a large group of receivers. Multicasting 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 buffer management schemes including a packet-discarding policy and retransmission control.

The standard method for providing a reliable unicast can be achieved by using positive acknowledgements (ACKs). This method requires the receiver to send an ACK for each packet that it has received. The sender keeps track of these ACKs and retransmits all packets that have not been properly acknowledged within a given time window. TCP [1] is a well-known protocol using positive ACKs to provide a reliable unicast.

The same approach fails when applied to reliable multicasts because of the ACK implosion [2]–[15] it creates. Since each receiver has to acknowledge each packet it has correctly received, the sender's ability to handle these ACKs limits the number of nodes participating in a reliable multicast.

This situation has led to numerous proposals aiming at providing scalable reliable schemes. The IETF Reliable Multicast Transport (RMT) Working Group has standardized three RMT protocols based on these proposals: asynchronous layered coding (ALC) [16], a negative acknowledgement (NAK)-based protocol [17], and tree-based protocols [3]–[12].

Among the protocols mentioned above, the tree-based protocols are known to provide high scalability as well as

reliability. They construct a logical tree at the transport layer for error recovery. 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 and controls the overall tree construction. Each repair node maintains in its buffer all the packets it has recently received and performs local error recovery for all its children nodes. As a result, tree-based protocols achieve scalability by distributing the server retransmission workload among the repair nodes.

There are still two open issues in tree-based protocols. The first is how to construct a logical tree in an efficient manner. One of the authors has recently proposed two efficient hybrid schemes for constructing a well-organized logical tree [3], [4]. Both schemes combine the advantages of previous schemes by constructing a logical tree in a semi-concurrent manner while minimizing the number of control messages.

The second open issue is when to discard packets from the buffers of repair nodes. Discarding packets that might still be needed is unacceptable because it would force the receiver nodes to contact its upstream node—which can be the sender node itself—whenever one of them needs a retransmission of a discarded packet. On the other hand, discarding packets too late would result in an inefficient use of the available buffer space on the repair nodes. Schemes addressing this issue can be broadly divided into ACK-based [8]–[12] and NAK-based schemes [6], [17]–[19]. As we will see, both approaches suffer from their own limitations.

In the ACK-based schemes, receiver nodes send an ACK to their repair node every time they have correctly received a packet. This allows each repair node to discard from its buffer all packets that have been acknowledged by all receiver nodes. However, the ACK-based approach does not scale up well due to the ACK implosion occurring at the repair nodes.

NAK-based schemes solve this ACK implosion problem by requiring the receiver nodes to send a NAK to their repair node each time they detect a packet loss. Unfortunately, they provide no efficient mechanism to safely discard packets from the repair node buffers. Hence, the repair nodes must discard the packets without knowing if the packets are still needed. Most tree-based protocols require these missing packets to be retransmitted by some upstream repair nodes. Unfortunately, these additional retransmissions can lead to a NAK implosion at the upstream repair nodes. Since the repair node cannot retransmit the requested packet immediately, these retransmissions also increase the error recovery delay. If the upstream 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 will often become unacceptable for many time-sensitive applications. Also, these additional retransmissions will increase the sender's workload.

We believe that the buffers of these repair nodes should be managed in an efficient manner because the unnecessary packets stored in their buffer waste storage resources. Moreover, if their buffer size is limited, the loss recovery time is increased.

We propose a heuristic approach to provide an efficient way to discard packets from the repair node's buffer. Under our scheme, each repair node predicts the packet that will not require retransmission and removes that packet from the buffer. Also, it 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 upstream repair node.

Our proposal has two major advantages over previous schemes. First, our heuristic reduces the amount of feedback from the receiver nodes. This feature provides scalability, since each repair node will be able to handle more receiver nodes. Second, our heuristic provides fast recovery of transmission errors since most of the packets requested by the receiver nodes are retransmitted from one of the members of their local group. This feature satisfies the original goal of tree-based protocols because each local group performs the error recovery by itself. It also reduces the size of the NAK implosion at the upstream repair nodes. As a result, our scheme provides an acceptable compromise between ACK-based and NAK-based schemes. In addition, it can be extended to provide flow control and eliminate any risk of buffer overflow.

The remainder of this paper is organized as follows. Section II summarizes the existing buffer management schemes for providing buffer refreshment functionality in a reliable multicast. Section III introduces our new buffer management 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 buffer management protocols of existing reliable multicast protocols. These protocols essentially differ in the strategies they use for deciding which participants should buffer packets for retransmission and how long these packets should be retained.

Scalable reliable multicast (SRM) [17] 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 of the

packets in their buffer for retransmission. Hence, the SRM protocol cannot provide an efficient buffer management mechanism at the transport layer.

The first tree-based reliable multicast protocol was the reliable multicast transport protocol (RMTP) [11]. RMTP provides reliable multicasting by constructing a physical tree of the network layer. It allocates a designated receiver in each local region and makes this receiver responsible for error recovery for all the other receivers in that region. To reduce the size of an 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 designated receiver for retransmission, which makes it poorly suited for transfers of large amounts of data. Some of these problems were addressed in RMTP-II [12] by the addition of NAKs.

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

Ozkasap [19] proposed an efficient buffering policy where only a small set of receivers buffer the packet to reduce the amount of total buffer requirements. Receivers that have not correctly received a given packet use a hash function to select the members that have the packet in their buffers and request a retransmission of the packet from one of them. Unfortunately, their selection method does not consider geographic locations between different receivers. Hence, its scalability is constrained because the latency for error recovery increases with the number of participants.

The randomized reliable multicast protocol [21] is an extended version of the bimodal multicast protocol [18]. The bimodal multicast protocol uses a simple buffer management policy in which each member buffers packets for a fixed amount of time. The randomized reliable multicast protocol uses 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 when the number of participants increases.

The search party protocol [20] 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.

We recently proposed a randomized scheme [5] requiring each receiver node to use both positive and negative acknowledgments to manage these buffers in an efficient manner. Under this scheme, the receiver nodes send negative acknowledgments to the repair nodes to request packet retransmissions. This NAK contains the sequence number of the last packet it has correctly received. At infrequent intervals, they also send randomized positive acknowledgments 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. It achieves this goal without increasing the server workload because (a) each receiver node only sends infrequent positive acknowledgments and (b) their sending times are randomized among all the receiver nodes. 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 randomization intervals for both receiver nodes and repair nodes.

Most of the 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. Yamamoto et al. [13], [22] have proposed an interesting flow-control scheme for NAK-based multicast protocols. Their scheme requires the sender to reduce its transmission rate whenever it receives NAKs for too many of its packets. The sender also keeps a log of its past transmission rates to prevent excessive rate decreases. While the scheme was found to be efficient, we should observe that it minimizes occurrences of buffer overflows rather than eliminating them, as a sliding window protocol would do.

III. Our Scheme

We propose a heuristic approach to provide an efficient way to discard packets from the repair node's buffer. Our heuristic scheme will allow most retransmissions to be handled by repair nodes, rather than the original sender node. Also, it minimizes

the number of feedbacks sent by receiver nodes. This feature increases the scalability and reduces recovery delay in the error recovery phase.

1. Buffer Refreshment

Our basic idea is that every group of receivers will include one or more members that experience higher error rates than the other members of the group. Hence, any packet that has been correctly received by these receivers will be likely to have been received by all the other members of the group. We also assume that most groups of receivers will include one or more members with slower links and that the NAKs returned by these members will almost always arrive to the repair nodes after those coming from the other members of the group. As a result, most NAKs coming from the other group members will likely reach the repair node before the NAKs coming from these slower receivers.

Our scheme requires each repair node to select one or more representative nodes from two distinct subsets of receiver nodes. First, the repair node will pick one or more representative nodes among the receivers having the highest loss probability. Then, it will select one or more additional representative nodes among the receivers having the slowest round-trip times. The repair node will discard packets from its buffer based on the feedbacks sent by these representative nodes.

In order to provide reliable multicasting, other schemes [11], [12], [23] have also proposed combining ACKs with NAKs. RMTP [11], [12] requires each receiver node to send ACKs at randomized intervals. The main characteristic of these ACKs is that they include NAKs. Hence, the repair nodes can be managed in a scalable manner, since they only deal with a reduced number of ACKs. However, RMTP fails to provide fast error recovery, because each receiver node will not request the lost packet immediately.

The pragmatic general multicast congestion control (PGMcc) protocol [23] also uses both ACKs and NAKs. It selects in each local group a specific node—called the *ACKer*—that acknowledges all packets while other nodes only send NAKs. Our scheme differs in two ways from PGMcc. First, PGMcc provides congestion control but does not address the issue of repair node buffer management. As a result, it selects as the ACKer of a group the node with the lowest throughput. Second, PGMcc uses a single ACKer per group, while our scheme allows for several representative nodes.

A. Picking Representative Nodes for Packet Discarding

In this subsection, we describe which receiver nodes are selected as representatives for their 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 nodes attached to a repair node has an independent packet loss probability, L_i, for $i = 1, 2, \ldots, N$.
- The repair node also has an independent loss probability, L_{rp}.
- Each of these receiver nodes has an independent NAK timer, NAK_TIMER_i.

The repair node calculates the packet loss probability of its N receiver nodes by counting the number of NAKs from each receiver node. Also, each receiver node sends its *NAK_TIMER* value to its repair node. Based on this information, the repair node selects $R_a = R_1 + R_2$ representative nodes as follows:

- Each repair node selects first the R_1 least reliable receiver nodes among the N receiver nodes it serves. These R_1 nodes will be the R_1 receiver nodes with the highest L_i for $i = 1, 2, \ldots, N$. Set $\mathcal{R}_1$ is the set containing these R_1 least reliable receiver nodes, and $|\mathcal{R}_1| = R_1$.
- Each repair node also selects the R_2 slowest receiver nodes. These R_2 nodes will be the nodes with the highest NAK delays times. Set $\mathcal{R}_2$ is the set containing these R_2 slowest receiver nodes, and $|\mathcal{R}_2| = R_2$.
- $\mathcal{R}_a = \mathcal{R}_1 \cup \mathcal{R}_2$ is the set of representatives.

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

These representative nodes will be asked to acknowledge all correctly received packets to their repair node. Our scheme uses these acknowledgments combined with the NAKs from the other nodes to decide when it is safe for a repair site to discard packets. Essentially, a repair node will keep its copy of a given packet until all the receiver nodes with the least reliable and the slowest connections have acknowledged its safe delivery. This will ensure that packets whose retransmission is requested by any other receiver node will almost always be available in the repair node's buffer.

We brought two modifications to this basic idea in order to increase the scalability of our scheme. First, we use *cumulative acknowledgments* to acknowledge all correctly received packets with ACK(n), meaning that the receiver node has correctly received up to packet n. Second, we delay these acknowledgments to let them better mimic the behavior of NAKs. The delayed acknowledgment (DAK) for representative node i will be delayed by a time-interval

$$OTT_{s,i} + S\_coeff \cdot NAK\_TIMER_i, \qquad i \in \mathcal{R}_a, \tag{1}$$

where $OTT_{s,i}$ is the one-way transit time between the sender

node and representative node i, NAK_TIMER_i is the NAK timer value specified by representative node i, and S_coeff is a safety coefficient.

Adding an extra one-way transit time, $OTT_{s,i}$, and multiplying the NAK_TIMER value by a safety coefficient will reduce the probability of having DAKs from representative nodes reach the repair node before NAKs from the other nodes. S_coeff takes into account (a) the fact that the delayed ACKs still can be returned faster than NAKs and (b) normal statistical variations.

Let $P(M_{Heuristic})$ represent the probability that a given packet was not correctly received by a node and could not be found in the buffer of its repair node. This event will occur whenever some receiver nodes request retransmission of packets that have already been discarded by the repair node because it had already received all the ACKs sent by the representative nodes. We adjust the S_coeff value to reduce $P(M_{Heuristic})$. This is done by applying the same multiplicative increase and additive decrease algorithm used in the recent TCP flow control protocol [13].

If the missing probability is larger than a predefined threshold, S_coeff is multiplied by factor k_s. The arrival time of delayed acknowledgment DAK_i to its repair node is then given by

$$OTT_{s,i} + S\_coeff \cdot NAK\_TIMER_i + OTT_{i,rp}, \quad i \in \mathcal{R}_a, \qquad (2)$$

where $OTT_{i,rp}$ is the one-way transit time between representative node i and its repair node rp.

Due to these adjustments, the DAK sent by a receiving node will almost always reach the repair node later than any NAK sent by the same node.

B. Packet Discarding

Each repair node will use the NAKs and DAKs sent by its representative nodes to predict which can be safely discarded from its buffer. Let LP_i be the last DAK packet sent by representative node i. Assuming that there is no pending packet retransmission, the repair node will now discard up to packet D such that

$$D \le \min\{LP_i \mid i \in \mathcal{R}_a\}. \qquad (3)$$

As a result, our scheme guarantees that the repair node will almost always have in its buffer all the packets that can be requested by any of its representative nodes.

Receiver nodes that leave a multicast session without any notice can disrupt the multicast session. The repair node will use a timeout mechanism to detect them and cut them off. Receiver nodes must also be able to handle the sudden loss of their repair node. Since this loss would leave the receiver nodes detached from the tree, they should immediately bind with the sender node and remain in that state until they can be properly reattached to the tree using the bind procedure described in our hybrid-tree-construction scheme [4] or some variant thereof.

Let us turn our attention to the probability that a repair node is not able to retransmit a requested packet. Let $P(M_{Heuristic})$ represent that probability.

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

$$\begin{aligned} P(M_1) &= P(\text{the repair node did not receive the packet}) \\ &\quad \times P(\text{some other nodes did not receive the packet and their NAKs were not lost}) \\ &= P(\text{the repair node did not receive the packet}) \\ &\quad \times (1 - P(\text{all receiver nodes either received the packet or did not receive the packet and their NAKs were lost})) \\ &= L_{rp}\left(1 - \prod_{i=1}^{N}(1 - L_i + L_i^{\,2})\right). \end{aligned} \qquad (4)$$

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, our safety coefficient S_coeff will ensure that some packets will still remain available if some NAKs arrive before the latest DAK. Let us call this probability A. The probability might be very close to 1 if the repair node has a large enough timer value or a large safety coefficient S_coeff. 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

$$\begin{aligned} P(M_2) &= P(\text{the repair node correctly received the packet}) \\ &\quad \times (1-A) \times P(\text{all representative nodes have received the packet and their ACKs were not lost}) \\ &\quad \times P(\text{some receiver nodes did not receive the packet}) \\ &\quad + P(\text{the repair node correctly received the packet}) \\ &\quad \times A \times P(\text{all representative nodes have received the packet and their ACKs were not lost}) \\ &\quad \times P(\text{all other nodes that did not receive the packet failed to notify the repair node}) \end{aligned}$$

$$
\begin{aligned}
&= (1-L_{rp})[(1-A)(\prod_{i\in\mathcal{R}_a}(1-L_i)^2)(1-\prod_{i\notin\mathcal{R}_a}(1-L_i)) \\
&\quad + A(\prod_{i\in\mathcal{R}_a}(1-L_i)^2)(\prod_{i\notin\mathcal{R}_a}(1-L_i+L_i^2)-(\prod_{i\notin\mathcal{R}_a}(1-L_i)))] \\
&= (1-L_{rp})[\prod_{i\in\mathcal{R}_a}(1-L_i)^2(1-\prod_{i\notin\mathcal{R}_a}(1-L_i)) \\
&\quad - A(\prod_{i\in\mathcal{R}_a}(1-L_i)^2)(1-\prod_{i\notin\mathcal{R}_a}(1-L_i+L_i^2)]
\end{aligned} \tag{5}
$$

As one can see, keeping A as close as possible to 1 will minimize the probability that a packet requested by a non-representative node will not be in the buffer of the repair node. The best way to achieve this is to select an appropriate delay for the DAKs sent by the representative nodes. Our algorithm achieves this goal by adjusting the value of its safety coefficient *S_coeff* value.

2. An Augmented Scheme

Let us now show how we can augment our scheme in order to let the local group handle even more packet retransmissions.

We can assume 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.

Our augmented scheme requires each repair node to select R_b trusted nodes in addition to the R_a representative nodes mentioned above. These R_b trusted nodes will be the R_b receiver nodes with the lowest packet loss probabilities L_i for $i = 1, 2, \ldots, N$. Set $\mathcal{R}_b$ is the set containing these R_b trusted nodes, and $|\mathcal{R}_b| = R_b$.

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

Whenever a repair node cannot satisfy a retransmission request with a packet from its own buffer, it will contact the trusted node that has the lowest loss probability. The trusted node will respond to the repair node with that packet if it has it. The repair node will then be able to retransmit the packet to the receiver node that requested it.

Two cases should be considered. First, the repair node might not have received the requested packet. In that case, the repair node will send a single NAK to its upstream repair node and hold onto the remaining NAKs for its receiver nodes, as it knows it will probably be able to service these requests from a trusted node without having to wait for the reply from its upstream repair node. The single NAK from the repair node should not be counted as an additional retransmission, as the upstream repair node does not make any distinction between repair nodes and receiver nodes. Since all other retransmission requests are handled within the local group, this feature eliminates NAK implosion problems at the upstream repair node.

In the second case, the repair node has already received and then discarded the requested packet. As in the first case, most retransmission requests will be handled within the local group. If we assume all trusted nodes have a long enough timer value for discarding packets, the packet-missing probability of our heuristic scheme can be given by

$$
\begin{aligned}
P(M_{Heuristic}) &= (P(M_1) + P(M_2)) \\
&\quad \times P(\text{no representative node received the packet}) \\
&= (P(M_1) + P(M_2)) \times \prod_{i\in\mathcal{R}_b} L_i \,.
\end{aligned} \tag{6}
$$

Even with $A = 1$, we need to consider the case when all the NAKs sent by non-representative nodes will be lost. In that case, the repair node will expel the missing packet before receiving a second NAK from one of the non-representative nodes that did not receive the packet. One possible solution is to let the repair node request the packet from its upstream repair node. Unfortunately, this solution cannot guarantee that this upstream 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. In addition, we need to consider that the upstream repair node serves receiver nodes that have more reliable and faster connections. The problem is that this upstream repair node could not have the packet in its buffer because its own estimation of the network condition could have let it select a shorter timer value than the current repair node. As a result, most of the requested packets could have to be re-sent by the sender node. This will result in unnecessary traffic thus decreasing the whole network performance.

We also need to mention that our scheme is different from the case using two or more statically assigned repair nodes in the local group. Our scheme uses instead a dynamic approach that periodically reselects new representative nodes to adapt to changing network conditions.

IV. Performance Analysis

In this section, we show the performance of the proposed buffer management scheme by computer simulation. In our simulator, we consider the case of one-to-many bulk data transfers, i.e., a multicast session consisting of one sender node and many receiver nodes.

Figure 1 shows the network topology we use in simulation. All the simulation experiments are performed for up to 100 receiver nodes per repair node. We assume that there are either 15 or 30 representative nodes in a local group, since we focus on the performance affected by the number of representative nodes.

We also need to consider the height of the repair tree. If the requested packet is not available in the repair node's buffer, the error recovery delay becomes more prominent as the height of the repair tree increases. In order to reach a reasonable evaluation of the error recovery delays of the proposed scheme and NAK-based scheme, 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 upstream repair nodes.

Tree-based multicast protocols require each receiver node to join the logical error recovery tree. A logical tree construction includes several steps: 1) advertising the multicast session, 2) discovering a repair node for each receiver node, and 3) binding a receiver node to the repair node. In the multicast session advertisement phase, all nodes obtain the multicast group address, the address of the sender node, and other necessary information for tree construction. This process can be realized by using a mechanism such as a web page announcement. After that, each receiver node starts to find one or more candidate repair nodes that are available in the session for its error recovery. Finally, each receiver node selects and binds to the best repair node, the one with the shortest time-to-live distance among the candidate repair nodes. This logical tree can be constructed using a bottom-up [7], [8], a top-down [9], or a hybrid scheme [3], [4]. We assume there are N_{rp} repair nodes and the repair nodes are pre-determined. This is the standard hypothesis made by all tree construction schemes [3], [4], [7]–[9].

Once they are attached to the tree, each receiver node must select its *NAK_TIMER* delay before requesting lost packets. In real networks, the underlying transport protocol needs to detect packet duplications especially in case of retransmission. Hence, a dynamic estimation algorithm for the NAK timer value should be provided for an 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 40 ms, which is an average value of the current round-trip time value.

1. Feedback Implosion

Under our scheme, most receiver nodes only send one NAK per incorrectly received packet. Only the few representative nodes acknowledge every packet. Over a session involving the transmission of m packets, the maximum number of feedbacks from its N receiver node will obey the inequality

$$F_{Heuristic} \le m(R_a + \sum_{i \notin \mathcal{R}_a} L_i), \tag{7}$$

where $\mathcal{R}_a$ is the set of representative nodes for the repair node.

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 $\Delta_{\min}$ between the numbers of feedbacks of the two schemes will be given by

$$\Delta_{\min} = m(N - R_a - \sum_{i \notin \mathcal{R}_a} L_i) .$$

Figure 2 shows how this difference increases as N increases when the repair node selects different percentages of receiver nodes as its representative nodes.

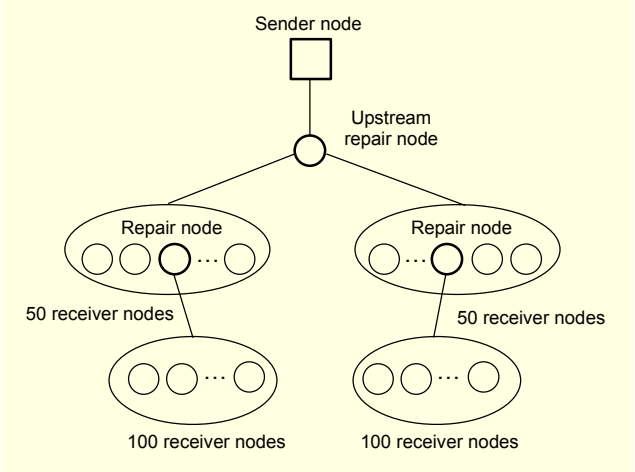

Fig. 1. Network topology.

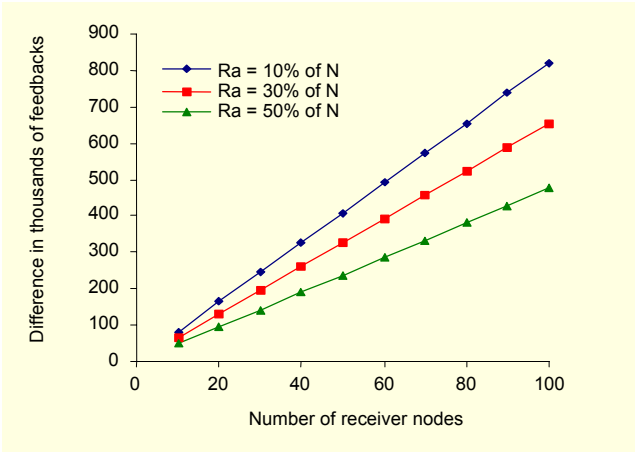

Fig. 2. Difference $\Delta_{\min}$ vs. the number N of receiver nodes per repair node.

We also assumed that the individual loss probabilities L_i are uniformly distributed between 0 and 1. We selected the number of transmitted packets as m = 10,000, which roughly represents a transfer of 10 megabytes with a packet size equal to 1 kilobyte. When there are 100 receiver nodes and the repair node selects 30 receiver nodes as its representative nodes, the minimum difference is about 650,000 feedbacks. This result indicates that our scheme provides efficient buffer management functionality for the repair node by reducing the number of feedbacks sent by the receiver nodes. This feature provides scalability since each repair node will be able to handle more receiver nodes.

2. Additional Retransmissions

Under our scheme, the repair node discards some packets based on ACKs sent by receiver nodes, which are on the least reliable and slowest connection. Hence, the repair node will have in its buffer most packets that can be requested by any of its receiver nodes. Also, the proposed scheme requires the repair node to request the missing packets to some representative nodes to be selected on the basis of their connection reliability. Therefore, the proposed scheme does not require many additional retransmissions either from its upstream repair nodes or sender node. These features provide fast error recovery for receiver nodes and a reduction in network traffic between the repair nodes.

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. Let us call the period δ and assume that the packets arrive at a repair node with a Poisson process with mean arrival rate λ. If the repair node has B buffers, we can define the random variable $N_A(\delta)$ to represent the number of packet arrivals at the repair node within a time interval of length δ. In order to perform at least one retransmission successfully, the following condition should be satisfied:

$$P(N_A(\delta) \geq B) = 1 - \sum_{n=0}^{B-1} \frac{(\lambda\delta)^n e^{-\lambda\delta}}{n!} = 0, \qquad (8)$$

which simplifies into $\sum_{n=0}^{B-1} \frac{(\lambda\delta)^n}{n!} = e^{\lambda\delta}$.

Since we have $e^{\lambda\delta} = \sum_{n=0}^{\infty} \frac{(\lambda\delta)^n}{n!}$, (8) can only be satisfied when B goes to infinity.

Hence, a NAK-based scheme 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.

In NAK-based schemes using a timer mechanism, the repair nodes discard packets from their buffers after a time interval I without considering whether these packets were received by all their receiver nodes. As a result, some packets might be removed from the repair node buffer while their retransmission could still be requested by one of the receiver nodes. In this case, the missing packets will have to be re-sent from either an upstream repair node or the sender node. In most cases, the packets will have to be re-sent by the sender node, especially when all repair nodes apply the same buffer management policy and discard the same packets at the same time. This generates unnecessary traffic, decreasing the whole Internet performance.

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

$$P(M_{NAK}) = L_{rp}\left(1 - \prod_{i=1}^{N}(1 - L_i + L_i^2)\right) + (1 - L_{rp})\left(1 - \prod_{i=1}^{N}(1 - L_i)\right)$$

for $A = 0$, and

$$P(M_{NAK}) = L_{rp}\left(1 - \prod_{i=1}^{N}(1 - L_i + L_i^2)\right) + (1 - L_{rp})\left(\prod_{i=1}^{N}(1 - L_i + L_i^2) - \prod_{i=1}^{N}(1 - L_i)\right)$$

for $A = 1$.

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

$$\begin{aligned} P(M_{Heuristic}) &= (P(M_1) + P(M_2)) \\ &\quad \times P(\text{no representative node received the packet}) \\ &= (P(M_1) + P(M_2)) \times \prod_{i \in \mathcal{R}} L_i \,. \end{aligned} \qquad (9)$$

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 1.

To generate the loss probability of each receiver node, we applied the formula $S = 1.22/(RTT_{s,i}\sqrt{L_i})$ from [24], 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. This assumes that the sender node transmits packets in a TCP-friendly manner and each node in the multicast session uses the UDP protocol.

Table 1. Configuration Parameters.

Sending rate S	128 packets/second
Avg_RTT	40 ms
Avg_OTT	15 ms
N	100 receiver nodes
R_a	15 and 30 representative nodes
R_b	3 representative nodes
L_{rp}	0.02
NAK_TIMER_i, $1 \le i \le N$	40 ms
Safety Coefficient	1
m	10,000 packets (≅10 Mbyte)

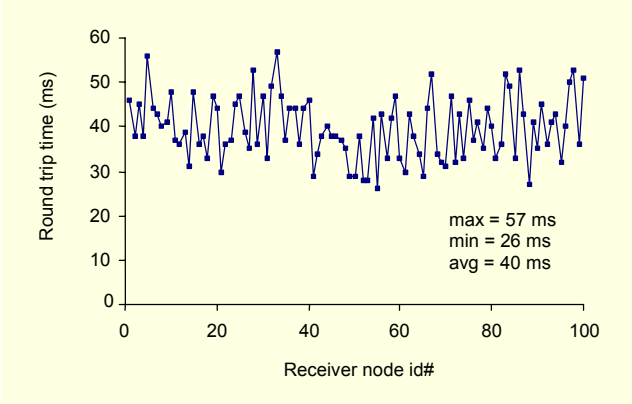

Fig. 3. Simulated round trip time.

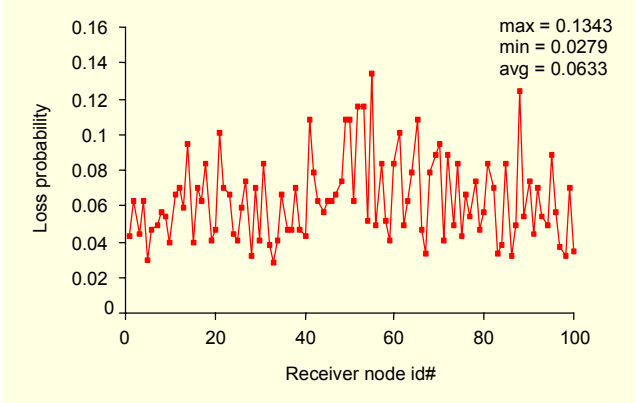

Fig. 4. Simulated loss probability.

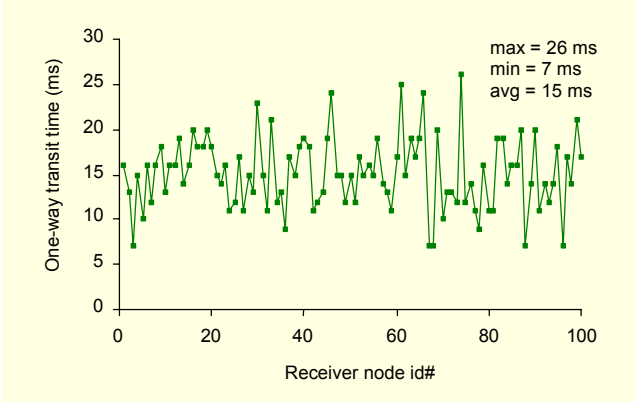

Fig. 5. Simulated one-way transit time.

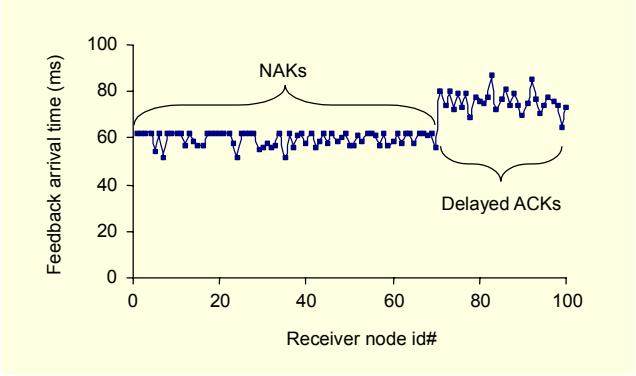

Fig. 6. NAK and DAK arrival times for each receiver node.

We simulated round-trip times $RTT_{i,j}$ 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 simulated by Poisson random variables with mean *Avg_OTT*. Figures 3, 4, and 5 respectively show our measurements for round trip times, packet loss probabilities, and one-way transit times for 100 receiver nodes.

Figure 6 shows the NAK and DAK arrival times for each receiver node when the repair node selects 30% of the receiver nodes as its representative nodes (20% for worst nodes and 10% for slowest node) and the propagation delay in the network is set up to 5 ms. The representative nodes correspond to the nodes numbered 71 to 100 in the figure. We can see that our delayed ACK mechanism ensures that acknowledgments from the representative nodes always arrive to the repair node after the NAKs from the other nodes. Hence, most of the requested packets from other nodes with NAKs will still be available in the repair node's buffer.

Using the configuration parameters in Table 1, we can evaluate 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 buffer for 120 ms. Recall that our scheme lets repair nodes discard packets once they have received DAKs from all representative nodes in R_a. The packets are then kept in the buffer of the R_b representative nodes for 40 additional milliseconds.

We assumed that the repair node selected three representative nodes for retransmission control, namely the node with the most reliable connection (the "best" node) and two "next best"

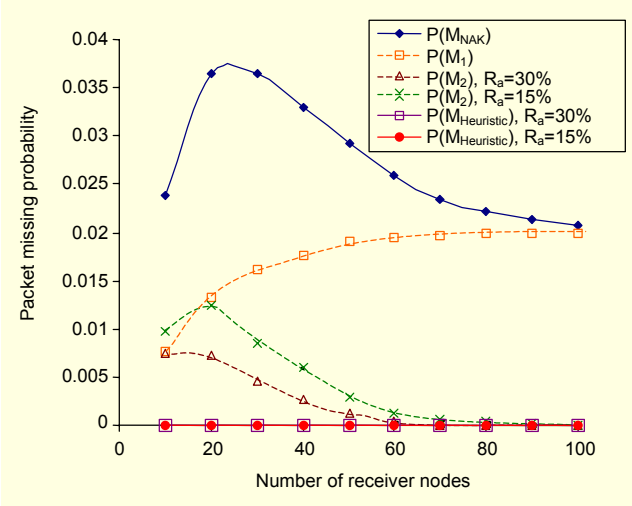

Fig. 7. Packet missing probability.

receiver nodes to act as a backup of the best node in case of a failed connection. As our probabilistic analysis will show, the result is acceptable even when the repair node has only one best receiver node as a representative node.

Figure 7 shows how the number of receiver nodes per repair node affects the probability of not finding a requested packet in the repair node buffer. We can see that the performance of the NAK-based scheme improves as the number of receiver nodes increases. Recalling that

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

for $A = 1$, we observe that the second term

$$(1-L_{rp})(\prod_{i=1}^{N}(1-L_i+L_i^2)-\prod_{i=1}^{N}(1-L_i))$$

decreases as the group size N increases. The contribution of this term is labeled as M_2 on the graph. Unfortunately, the first term (M_1)

$$L_{rp}(1-\prod_{i=1}^{N}(1-L_i+L_i^{\,2}))$$

tends to L_{rp} when the same group size N increases. As a result, the packet-missing probability for the NAK scheme will always be greater than or equal to L_{rp}. As a result, the NAK-based scheme will always perform significantly worse than our scheme despite having a larger timer delay.

We can also see that our heuristic scheme always achieves very low packet-missing probabilities for all numbers of receiver nodes considered.

This probability is about 10^{-10} when there are 100 receiver nodes including 30 R_a type representative nodes (20 worst receiver nodes and 10 slowest receiver nodes) and 3 R_b type representative nodes (one best receiver node and two backup receiver nodes).

We should also mention that the conditions under which the comparison is performed are unfavorable to our scheme as we assumed that all receiver nodes had an identical failure rate and transmission delay distributions. In most real situations, some receiver nodes will either experience longer transmission delays or less reliable links. Selecting these worse receiver nodes as representatives will provide even lower missing packet probabilities without all the disadvantages of having to select a very large timer delay.

The performance of the NAK-based scheme will improve whenever the repair nodes have very large buffers as well as 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 a buffer overflow sooner or later.

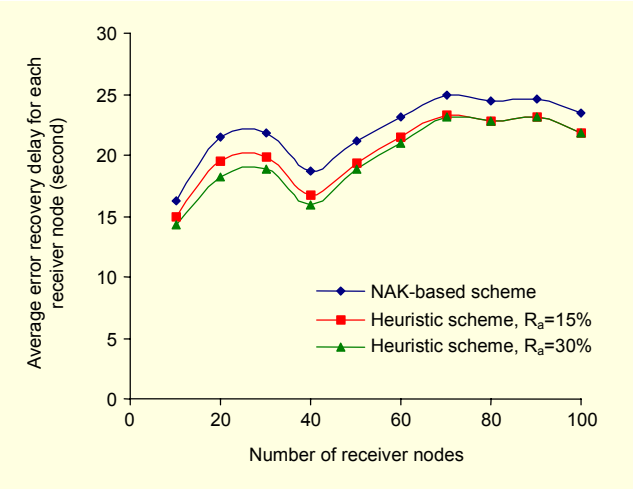

Fig. 8. Difference between the error recovery delays.

3. Estimating the Error Recovery Delay

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 some upstream repair node. This might double or triple the error recovery delay. Also, these additional retransmissions cause unnecessary traffic between the repair nodes.

To evaluate the minimum delay difference between both

schemes, we assume that the additional retransmission is always correctly transmitted from the upstream repair node. Otherwise, the difference would be much more prominent. We also assume that the round trip time between the two repair nodes is 30 ms.

Under these assumptions, each receiver node measures the average recovery delay over all the packet loss it experienced. The results are shown in Fig. 8.

Even under simulated parameter values, the proposed scheme performs better than the NAK-based scheme. Note that the difference decreases as the number of receiver nodes per repair node increases since the probability of finding an extremely slow node among a larger number of receiver nodes increases. This is to some extent an artifact of our model because we model transmission delays by unbounded Poisson variables while actual transmission delays are bounded.

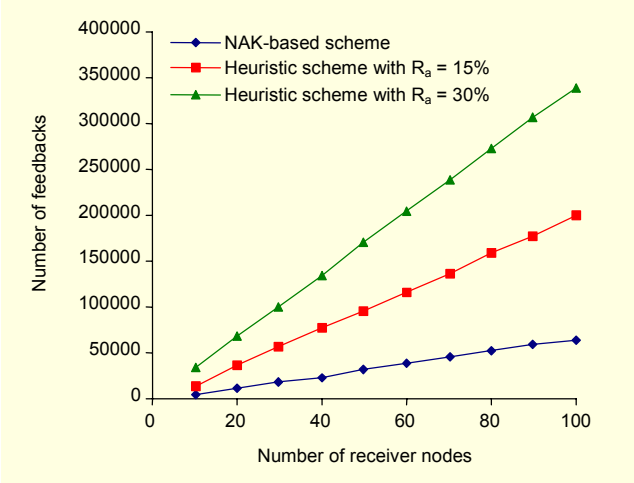

Fig. 9. Difference between the numbers of feedbacks per repair node.

4. The Number of Representative Nodes

As we can see in Figs. 8 and 9, the number of representative nodes selected by the repair node slightly affects the overall performance of the proposed scheme.

There is a tradeoff between the number of feedbacks sent by the receiver nodes and the number of additional retransmissions. Hence, the repair node needs to limit the number of representative nodes by considering how many feedbacks it can handle.

V. Conclusion

We have proposed a heuristic buffer management scheme combining NAKs and delayed ACKs to provide scalability and reliability in a multicast session. Under our scheme, most receiver nodes send only negative acknowledgments to repair nodes to request packet retransmissions. Our scheme selects a few receiver nodes among the nodes with the slowest links and those with the least reliable links. These receiver nodes send delayed ACKs to their repair nodes to indicate which packets can be discarded by the repair node. Our improved scheme also selects a few receiver nodes with the most reliable links and requires them to keep a copy of all received packets for a finite interval of time. Whenever the repair node cannot satisfy a retransmission request from its own buffer, it will request the missing packet from one of these trusted nodes rather than from its upstream repair node. These representative nodes send the requested packet to the repair node if they have the packet. Hence, more nodes participate in error recovery, which provides a fast recovery of the transmission error.

Our scheme shows a better performance than an ACK-based scheme in terms of the number of feedbacks sent by the receiver nodes. In addition, the number of additional retransmissions is reduced compared to a NAK-based scheme. Hence, it provides an acceptable trade-off between ACK-based and NAK-based schemes, using both positive and negative acknowledgments to achieve reliability and scalability.

References

[1] M. Mathis, J. Mahdavi, S. Floyd, and A. Romanow, "TCP Selective Acknowledgment Options," RFC 2018, Oct. 1996.

[2] R. Yavatkar, J. Griffieon, and M. Sudan, "A Reliable Dissemination Protocol for Interactive Collaborative Applications," *Proc. of the 3rd ACM Int'l Conf. on Multimedia*, San Francisco, USA, Nov. 1995, pp. 333-344.

[3] J. Baek, "A Hybrid Configuration of ACK Tree for Multicast Protocol," *Proc. of the 2002 Int'l Symp. on Performance Evaluation of Computer and Telecommunication Systems* (*SPECTS 2002*), San Diego, USA, July 2002, pp. 852-856.

[4] J. Baek, "An Improved Logical Tree Construction Scheme for Tree-Based Reliable Multicast," *Proc. of the 2003 Int'l Conf. on Telecommunication Systems* (*ICTS 2003*), Monterey, USA, Oct. 2003, pp. 110-121.

[5] J. Baek and J.F. Paris, "A Buffer Management Scheme for Tree-Based Reliable Multicast Using Infrequent Acknowledgments," *Proc. of the 23rd IEEE Int'l Performance Computing and Comm. Conf.* (*IPCCC 2004*), Phoenix, USA, Apr. 2004, pp. 13-20.

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

[7] C. Maihofer and K. Rothermel, "A Robust and Efficient Mechanism for Constructing Multicast Acknowledgment Trees," *Proc. of the 8th IEEE Int'l Conf. on Computer Comm. and*

*Network*s, Boston-Natick, USA, Oct. 1999, pp. 139-145.

[8] K. Guo and I. Rhee, "Message Stability Detection for Reliable Multicast," *Proc. of the 19th IEEE Conf. on Computer Comm.* (*INFOCOM 2000*), New York, USA, Mar. 2000, pp. 814-823.

[9] S.J. Koh, E. Kim, J. Park, S.G. Kang, K.S. Park, and C.H. Park, "Configuration of ACK Trees for Multicast Transport Protocols," *ETRI J.*, vol. 23, no. 3, Sept. 2001, pp. 111-120.

[10] M. Kadansky, B. Whetten, B. Cain, D.M. Chiu, B. Levine, D. Thaler, S. Koh, and G. Taskale, "Reliable Multicast Transport Building Block: Tree Auto-Configuration," *IETF Internet Draft*, draft-ietf-rmt-bb-tree-config-01.txt, Nov. 2000.

[11] J.C. Lin and S. Paul, "RMTP: A Reliable Multicast Transport Protocol," *Proc. of the 15th IEEE Conf. on Computer Comm.* (*INFOCOM '96*), San Francisco, USA, Mar. 1996, pp. 1414-1424.

[12] B. Whetten and G. Taskale, "The Overview of Reliable Multicast Transport Protocol II," *IEEE Networks*, vol. 14, no. 1, Jan.-Feb. 2000, pp. 37-47.

[13] K. Yamamoto, M. Yamamoto, and H. Ikeda, "Performance Evaluation of ACK-Based and NAK-Based Flow Control Mechanisms for Reliable Multicast Comm.," *IEICE Trans. on Comm.*, vol. E84-B, no. 8, Aug. 2001, pp. 2313-2316.

[14] B. Levine and J.J. Garcia-Luna-Aceves, "A Comparison of Reliable Multicast Protocols," *ACM Multimedia Systems J.*, vol. 6, no. 5, Aug. 1998, pp. 334-344.

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

[16] M. Luby, L. Vicisano, J. Gemmell, L. Rizzo, M. Handley, and J. Crowcroft, "The Use of Forward Error Correction in Reliable Multicast," IETF draft-ietf-rmt-info-fec-02.txt, Oct. 2002.

[17] S. Floyd, V. Jacobsen, C.G. Liu, S. McCanne, and L. Zhang, "A Reliable Multicast Framework for Lightweight Sessions and Application-Level Framing," *IEEE/ACM Trans. on Networking*, vol. 5, no. 6, Dec. 1997, pp. 784-803.

[18] K.P. Birman, M. Hayden, O. Ozkasap, Z. Xiao, M. Budiu, and Y. Minsky, "Bimodal Multicast," *ACM Trans. on Computer Systems*, vol. 17, no. 2, May 1999, pp. 41-88.

[19] O. Ozkasap, R. van Renesse, K.P. Birman, and Z. Xiao, "Efficient Buffering in Reliable Multicast Protocols," *Proc. of the First Int'l Workshop on Networked Group Communication* (*NGC'99*), Pisa, Italy, Nov. 1999, pp. 188-203.

[20] M. Costello and S. McCanne, "Search Party: Using Randomcast for Reliable Multicast with Local Recovery," *Proc. of the 18th IEEE Conf. on Computer Comm.* (*INFOCOM '99*), New York, USA, Mar. 1999, pp. 1256-1264.

[21] Z. Xiao, K.P. Birman, and R. Renesse, "Optimizing Buffer Management for Reliable Multicast," *Proc. of the Int'l Conf. on Dependable Systems and Networks* (*DSN'02*), Washington, D.C., USA, June 2002, pp. 187-202.

[22] M. Yamamoto, Y. Sawa, S. Fukatsu, and H. Ikeda, "NAK-Based Flow Control Scheme for Reliable Multicast Communications," *IEICE Trans. on Comm.*, vol. E82-B, no. 5, May 1999, pp. 712-720.

[23] L. Rizzo, "PGMcc: a TCP-Friendly Single-Rate Multicast Congestion Control Scheme," *Proc. of the ACM SIGCOMM 2000*, Stockholm, Aug. 2000, pp. 17-28.

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

Jinsuk Baek received the BS and MS degrees in computer science and engineering from Hankuk University of Foreign Studies (HUFS) in Yougin, Korea, in 1996 and 1998 and the PhD in computer science from the University of Houston in 2004. Dr. Baek is a member of the Distributed Multimedia Research Group at the University of Houston. His research interests include scalable reliable multicast protocols, mobile computing, proxy caching systems and formal verification of communication protocols. He is a member of IEEE.

Jehan-François Pâris is a Professor of computer science at the University of Houston. He was formerly on the faculty at Purdue University and the University of California, San Diego. Dr. Pâris received his *Ingénieur Civil* degree from the Université Libre de Bruxelles, Belgium, his *Diplôme d'Etudes Approfondies* from the Université de Paris VI, France, his *Licence et Maîtrise en Informatique* from the Facultés Universitaires de Namur, Belgium and his PhD in EECS from the University of California, Berkeley. His research interests include memory hierarchies, scalable reliable multicast protocols, distribution protocols for video-on-demand, and distributed systems security. He is a member of ACM and a senior member of IEEE.