Packet Loss Patterns Adaptive Feedback Scheduling for Reliable Multicast

Jinsuk Baek*, Cheonshik Kim**, and You-Sik Hong***

Abstract — **Tree-based reliable multicast protocols provide scalability by distributing error-recovery tasks among several repair nodes. These repair nodes perform local error recovery for their receiver nodes using the data stored in their buffers. We propose a packet loss patterns adaptive feedback scheduling scheme to manage these buffers in an efficient manner. Under our scheme, receiver nodes send NAKs to repair nodes to request packet retransmissions only when the packet losses are independent events from other nodes. At dynamic and infrequent intervals, they also send ACKs to indicate which packets can be safely discarded from the repair node's buffer. Our scheme reduces delay in error recovery because the requested packets are almost always available in the repair node's buffers. It also reduces the repair node's workload because (a) each receiver node sends infrequent ACKs with non-fixed intervals and (b) their sending times are fairly distributed among all the receiver nodes.**

Index Terms — **Reliable Multicast, Temporal locality, Spatial locality, Feedback schedule**

I. INTRODUCTION

This A growing number of network applications, such as bulk data transfer, distance learning, video-conferencing, data feeds, Internet TV, Web cache update, and distributed interactive gaming, require a sender to distribute the same data to a large group of receivers.

These applications fall in the category of group communication as opposed to traditional one-to-one communication. Multicasting, that is the delivery of a single message to multiple recipients using the same IP address, is the only efficient and scalable solution to support this kind of application.

One of the most difficult issues in end-to-end multicasting is that of providing an error-free transmission mechanism [4]. IP multicast does not guarantee reliable packet delivery as it uses best efforts transmission mechanism. Hence, lost or damaged packets should be retransmitted from the sender or some other node. This process can be performed at the transport layer.

The standard method for providing reliable unicast is positive acknowledgements (ACKs). It 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 [3]. TCP [13] is a well-known protocol using positive ACKs to provide reliable unicast. The same approach fails when applied to reliable multicasts because of the ACK implosion [1]–[12], [14] 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.

Among the many protocols [1]–[12], [14], most recently, overlay reliable multicast protocols, controlling and maintaining an efficient overlay for data transmission, have been proposed. However, there are still many issues to be considered including addressing and advertisement, path quality measurement, tree refinement, and failure recovery. In this paper, we focus on the tree-based protocol that has been known to provide high scalability and reliability. They construct a logical tree at the transport layer for error recovery. This logical tree comprises of three types of nodes: a sender node, repair nodes, and receiver nodes. The sender node is the root of the 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 group member nodes. As a result, tree-based protocols achieve scalability by distributing the server retransmission workload among the repair nodes.

In the tree-based protocols, one of the most important open issues is when and how frequently the receiver nodes have to send ACKs to their repair node in a local group. The ACK for every successfully received packet causes ACK implosion at the repair node while the ACK with a long interval definitely delays the repair node's buffer refreshment timing. On the other hand, NAK-based approaches [1], [6]–[8], [10] solve the ACK implosion problem by shifting the error detection load from the repair nodes to the receiver nodes. Receiver nodes will now reply with negative acknowledgments (NAKs) whenever they detect packet losses. Unfortunately, it is very difficulty to decide when the repair node can safely discard packets from its buffers.

To the best of our knowledge, all previous solutions, focused on the ACK implosion issue, have their own limitations because they assumed that a) packet losses were independent events that were not correlated with previous transmission failures [1]–[4], [7]–[12], [14] or b) packet losses tend instead to happen in bursts [5], [6]. However, we need to point out we have to consider both cases for providing more efficient error recovery. Even though today's link is fairly reliable, link error still exists. Hence, the packet losses can be independent events among all nodes in the same local group. On the other hand, it also can be strongly correlated because packets losses are likely to be occasioned by router buffer overflows. Therefore, all group members can experience packet losses for the same packet.

We propose an efficient feedback scheduling scheme eliminating all of these limitations by using packet loss pattern adaptive ACKs. Our mechanism allows receiver nodes to send

Manuscript received Mar. 10, 2007 ; revised Aug. 15, 2007.

* 601 M.L.King Jr Dr, Department of Computer Science, Winston-Salem State University, Winston-Salem, NC 27110, USA

** 708-113 Anyang 5-Dong, Manan-Gu, Anyang-Shi, Kyeonngi-Do, Major in Digital Media Engineering Dept., Anyang University, South Korea

*** 660 Woosan-dong, Wonju-Shi, Kangwon-Do, Department of Computer Science, Sangji University, South Korea

NAKs to their repair node only when the packet losses are the independent event. At dynamic infrequent intervals we observantly defined, they also send ACKs to indicate which packets can be safely discarded from the repair node buffers. Whenever the repair node detects continuous packet losses, it multicasts a NAK suppression message to its receiver nodes.

Our proposal has many advantages over other schemes. First, we avoid any risk of ACK implosion because each receiver node only sends infrequent ACKs and its sending times are well distributed among all the receiver nodes. This feature provides scalability, since each repair node will be able to handle more receiver nodes. Second, it guarantees fast recovery of transmission errors, since the packets requested from receiver nodes are always available in the buffers of the repair nodes. Finally, we also can avoid NAK implosion for the same packet at the repair node because the repair node autonomously detects the high potential spatial locality of the packet losses among the group members. After receiving the *NAK suppression* message from the repair node, the receiver nodes will refrain from sending NAKs for every lost packet.

The remainder of this paper is organized as follows: Section 2 summarizes the existing feedback scheduling schemes for providing buffer refreshment functionality of the repair node. Section 3 introduces our new feedback scheduling scheme. In section 4, we show the performance of the proposed scheme. Finally, section 5 contains our conclusions.

II. Related Work

Although all tree-based multicast protocols strive for the same goals, they essentially differ in how frequently each receiver node sends their feedbacks to their repair node and also how long time the repair node to optimally retain these packets without compromising buffer space.

Scalable Reliable Multicast (SRM) [8] is a well-known NAK-based multicast protocol that guarantees out-of-order reliable delivery using NAKs from receivers. Every time a receiver detects a lost packet, it multicasts a NAK message to all participants in the multicast group. Unlike the regular NAK [1] where each individual receiver independently sends a NAK to the sender as soon as it detects a loss, the SRM protocol requires that each receiver use a randomized NAK-Timer to send the NAK. This allows the nearest receiver, among those that successfully received the packet, to retransmit it by multicasting. This is a very common NAK suppression scheme. Unfortunately, this requires all receiver nodes to indefinitely retain all of their successfully received packets for eventual retransmissions, consequently leading to poor buffer management. But the scheme would only work well if it were guaranteed that at least one receiver node in the local group would successfully receive the packet. This case can never be guaranteed especially when considering spatial locality of packet losses in multicasting.

Reliable Multicast Transport Protocol (RMTP) [12], the first tree-based reliable multicast protocol, selects a *Designated Receiver* (DR) that will be responsible for error recovery for all the other receivers in the region. Instead of sending an ACK for every packet received, each receiver periodically sends an ACK to the designated receiver. This ACK contains the maximum packet number that each receiver has successfully received. Unfortunately, this periodic ACK, without considering temporal locality of the packet losses for a receiver node, significantly increases error recovery delay since the receivers do not immediately request for retransmission for lost packets even though it has experienced continuous packet losses. This renders RMTP unfavorable in time-sensitive multimedia data applications. Moreover, since RMTP stores the whole multicast session data in the secondary memory of the DR for retransmission, it makes it unfavorable for transfers of large amounts of data. Some of these problems were addressed in RMTP-II [14] by the addition of NAKs.

The author has recently proposed several buffer management schemes: The first scheme [3] suggests that each receiver node send NAKs to its repair node for all its packet retransmission needs. At the same time this receiver also periodically sends randomized sequence numbered ACKs back to the repair node to signal for a safe discard from this repair node, which in turn observes the least packet sequence number among these packets and discards all packets with a sequence number below this minimum number. However, still not explored here is the optimal ACK transmission interval for both the repair and the receiver nodes. In addition, these ACKs, sent by at a fixed interval, are not adaptive to the temporal and spatial locality of the packet losses. The second scheme [4] suggests for the repair node to discard some packets by considering the ACKs from the most unreliable receivers. This scheme works well as long as there are some unreliable or very reliable receiver nodes compared to other nodes in a local group. In both schemes, there is a reduction in error recovery delay since a request for a packet from the repair node is almost always satisfied. Both schemes also minimize the number of repair nodes needed per number of receiver nodes by minimizing ACK implosion. It goes without noticing, however, that neither of these schemes addresses how to solve the NAK implosion problem and the locality nature of the packet losses.

This problem was handled in our most recent scheme [6] with some partial successful achievement. Under the scheme, the repair node broadcasts a *NAK suppression* message to all its group members if it detects the packet loss. Simultaneously, it will request a retransmission of that packet to the original sender node. If the receiver node detects a packet loss, but did not receive the suppression message for the packet, it multicasts a *NAK suppression* message for this packet to its group members including the repair node. If other receiver nodes already had the packet or received the *NAK suppression* message, it should ignore the new suppression message. In both cases, other receiver nodes will not send any NAK for this packet. When the repair node receives the *NAK suppression* message of a packet that it already has, it will multicast the packet to group members. Worth noting here is that in some cases, where all nodes are under the same router and the NAK timer of the receiver node is shorter than that of the repair node, the receiver nodes are bound to send NAK to the repair node before the *NAK suppression* message from the repair node reaches them. The result of this is that the receiver node will receive the *NAK suppression* message from the repair node after they have already sent a NAK to this repair node. In order to handle this problem, the scheme employed a random NAK sending delay for the receiver nodes to ensure that they always send a NAK later than their *NAK suppression* message reaches them from the repair node.

Although this scheme produced better results compared to the existing ones, it has some limitations in some specific cases. The first case is where the repair node and some receiver nodes successfully receive the packet but other receiver nodes experience a loss of this packet. Although the packet is available at the repair node and could be retransmitted right away, these receivers will still have to obey their delayed NAK timer value wait-time before they can send their NAK to the repair node. The result of this is unnecessary delay in the error recovery. The second scenario is where all the receivers of a local area successfully receive the packet but their associated repair node suffers a loss for that packet. This scenario can happen in a rare event if we consider the independent packet losses caused by link error. The repair node will multicast a *NAK suppression* message to these receivers although they successfully received the packet. This will lead to duplication of packets, and in so doing it will also cause unnecessary traffic.

III. PROPOSED SCHEME

We assume a receiver-initiated error recovery process and require receiver nodes to send feedbacks, such as ACK and NAK, to their repair node. In addition, we make the following assumptions:

1. The logical tree is already constructed using a well known tree construction scheme [2], [9], [11] before the multicast session begins. Hence, the repair node for every local group of receiver nodes is strategically allocated.
2. Ts is the multicast session period.
3. There are n receiver nodes in a local group and the repair node has packet retransmission responsibility for its n receiver nodes.
4. Each receiver node i has well defined its independent NAK_TIMERi satisfying the following condition.
   $NAK\_TIMER_i > \max\{OTT_{<s,i,t>} \mid 1 \le i \le n \text{ and } 0 < t < Ts\}$.
   where, $OTT_{<s,i,t>}$ is one-way transit time between the sender s and receiver node i at time t.
5. The repair node also has well defined NAK _TIMERrp, satisfying the following condition.
   $NAK\_TIMER_{rp} > \max\{OTT_{<s,rp,t>} \mid 0 < t < Ts\}$.
   where, $OTT_{<s,rp,t>}$ is one-way transit time between the sender s and repair node rp at time t.

When a receiver node i joins a multicast session, it receives a packet from its repair node specifying the maximum number RP_{max} of sent packets the repair node is normally willing to keep in its buffer. The receiver node i generates then a random number RC_{i_rand} between 1 and RP_{max} that identifies the first packet after which it will send an ACK packet to its repair node. This packet will act as an implicit ACK for packets 1 to RC_{i_rand}. After that, each receiver node i will send a feedback for the packet RC_i such that

$$RC_i = \begin{cases} 0 & \text{initialization} \\ RC_i + RC_{i\_rand} & \text{start and restart} \\ RC_i + I & \text{if no packet loss within the interval } I \\ RC_{i\_last} & \text{if any packet loss within the interval } I \end{cases}$$

The interval I is given by

$$I = \begin{cases} RP_{max} + 1 & \text{initialization} \\ I + 1 & \text{if no packet loss} \end{cases}$$

That is, if there is no packet loss before sending ACK(RC_{i_rand}), the receiver node i increases its interval by one and sends a similar ACK packet after each packet RC_i. On the other hand, if there is a packet loss within the interval, it sends a NAK for the lost packet RC_{i_lost}. Since ACK transmissions for each receiver node i start at random offsets

$$RC_{i\_rand} = \text{random}(1, RP_{max}) \text{ for } 1 \le i \le N, \quad (1)$$

they will be more or less equally distributed over time if the link is fairly reliable. Therefore, we eliminate the risk of a sudden avalanche of ACKs sent for the same packet. Let L_i be the last ACK packet sent by receiver node i. Assuming that there is no pending packet retransmission, the repair node now can safely discard up to packet D such that

$$D \le \min\{L_i \mid 1 \le i \le N\} \quad (2)$$

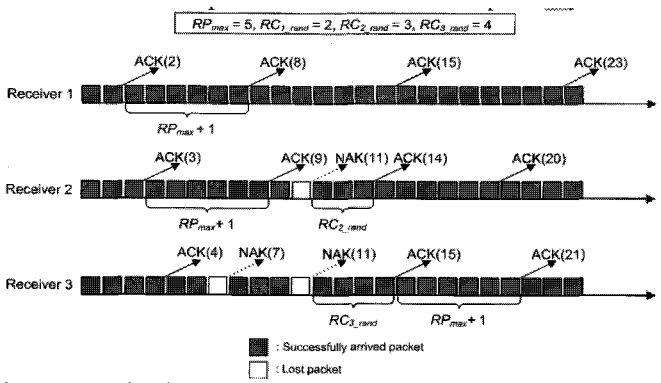


Fig. 1. Example of feedback scheduling for our scheme (with $RP_{max} = 5$).

Let us turn now our attention to the way our scheme handles lost packets. If the repair node detects back to back lost packets, it multicasts a *NAK suppression* message to its all group members. At the same time, it requests retransmission of the lost packets to its sender node rather than upstream repair node. In this case, the proposed scheme will avoid the existing scheme fault of always contacting the adjacent upstream repair node. By cotacting the root sender directly, a repair node in the proposed scheme will be able to bypass not only same level link error packet loss occurrences but also the worst case where the router adjacent to the root experiences a buffer overflow.

If it receives a NAK from one or some of the receiver nodes and it has the packet, it retransmits the packet to the receiver nodes who requested the retransmission. When a receiver node *i* detects a lost packet and it did not receive any *NAK suppression* message for that packet, it sends a NAK to its repair node. Note that this NAK will contain the sequence number of the last packet such that itself and all its predecessors were correctly received by receiver node *i*. After that, it restarts its ACK sending schedule. It is helpful for the repair node to perform a fast packet discarding. For example, all other receiver nodes in a group successfully received the packet because the packet loss was caused by the independent link error of the receiver node. In this case, other nodes will send ACKs according to their scheduled intervals which are most probably relatively longer than the receiver node which restart its ACK sending schedule. If it receives the *NAK suppression* message from the repair node, it stops to send NAKs and restarts to send ACKs with its most current interval. In this case, all lost packets within the range of spatial locality are delivered by its repair node. Fig.1 shows an example of our feedback scheduling for the four receiver nodes having different packet loss patterns when RP_{max} is equal to 5.

The first main advantage of our scheme is that it significantly reduces the number of feedbacks required to safely discard the packets from repair node's buffer. This feature increases the whole Internet performance by reducing unnecessary traffic.

A second advantage of our scheme is that it guarantees that the repair node will almost always have in its buffer all the packets that can be requested by any of its receiver nodes. We achieve this reliability using cumulative ACK mechanism. As we mentioned above, our ACK(n) means the receiver has correctly received up to packet n. Assume, for instance, that a receiver node i has not received packet k. It will send a NAK(k) to the repair node as long as it is an independent event. The repair node keeps in its buffer all packets with sequence numbers greater than the last acknowledged packet L_i and $L_i < k$. Hence, packet k is always available in repair node's buffer and can be retransmitted upon request. In addition, the arrival time of the next acknowledgment packet ACK(L_i') is always larger than that of NAK(k) because the receiver node i cannot send an ACK(L_i') until it receives up to packet L_i'. It means the receiver node i does not send ACK(L_i') until after it receives packet k. This property can be described as

Table 1. A summary of buffer management schemes

Schemes	Category			Advantage/Weakness	
	Tree based	ACK based	NAK based	Advantage	Weakness
SRM	No	No	Yes	· No ACK implosion. · NAK implosion is reduced owing to the NAK suppression.	All nodes have to keep all of the packets they have received.
RMTP	Yes	Yes	No	ACK implosion is reduced owing to its periodic feedbacks sent by receiver nodes.	· Long error recovery delay due to the periodic feedbacks. · No consideration of packet loss locality.
Heuristic	Yes	Yes	No	· ACK implosion is reduced because only the limited numbers of receiver nodes have to send ACKs. · Fast error recovery delay.	No consideration of packet loss locality.
Fixed interval	Yes	Yes	No	ACK implosion is reduced owing to its periodic ACKs with a fixed interval.	· No consideration of packet loss locality. · Difficult to find the optimal interval.
NAK Supp.	Yes	No	Yes	· No ACK implosion · NAK implosion is reduced owing to the NAK suppression. · Consideration of packet loss locality.	Unnecessary error recovery delay due to their NAK timer.
Dynamic interval	Yes	Yes	No	· ACK implosion is reduced owing to its periodic ACKs with a dynamic interval. · Consideration of packet loss locality. · Less buffer space.	Error recovery delay with extremely rare event.

Arrival_Time(ACK(L_i')) > Arrival_Time(NAK(k)),

because $L_i' > k$. As a result, the proposed scheme reduces the packet error recovery delay because no packet will ever have to be retransmitted either by the original sender node. The error recovery delay is only dependent on the round-trip time between the repair node and the receiver node requesting the retransmission of the packet. Finally, our feedback scheduling is adaptive to the given packet loss patterns. The *NAK suppression* message sent by repair node allows the repair node to avoid NAK implosion for the continuous packet losses because each receiver node is forced to stop to send NAKs for every lost packet. Also, our scheme allows relatively unreliable receiver node to sends more ACKs because they will more frequently restart their ACK sending schedule from the initial phase.

3.1 Comparison of Schemes

In the pure ACK-based scheme, each receiver node sends independent ACKs for every successfully received packet to their repair node. This simply causes an ACK implosion at the repair node. In order to reduce the ACK implosion, ACK-based scheme with a fixed interval allows each receiver node to send periodic ACKs. However, it is very complicate problem to decide the fixed interval because the buffer status for the repair node is likely to be changed over time. On the other hand, heuristic scheme only allow a few very unreliable receiver node to send ACKs. Based on these ACKs, the repair node can perform packet discarding. However, this does not always work well because sometimes reliable nodes can experience packet losses for the packet that the unreliable nodes have successfully received. In order to handle this scenario, some of the reliable nodes should keep the packets for some amount of time to perform packet retransmission.

The NAK-based schemes eliminate ACK implosion by allowing each receiver node to send NAKs for only incorrectly received packets. However, it is difficult and too complicate to decide the optimal packet discarding timing at the repair node. As we will prove in the next section, in order to avoid unnecessary additional retransmissions, the repair node theoretically has to keep all of the received packets until the multicast session ends.

The proposed scheme is an ACK-based scheme having a dynamic interval. This dynamism is based on the different packet loss patterns observed at each receiver node. The comparison of various schemes is summarized in Table 1.

IV. PERFORMANCE

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

4.1 Feedback Implosion

The first main advantage of our scheme is that it significantly reduces the number of feedbacks sent by receiver nodes, required to safely discard the packets from repair node's buffer. We evaluate the number of feedbacks sent by receiver nodes and compare the result to the existing ACK-based scheme – namely randomized scheme having a fixed interval [3].

Over a session involving the transmission of m packets and the receiver node i has the packet loss probability LP_i, the maximum number of feedbacks from receiver node i for the randomized scheme with a fixed interval is given by

$$\lceil m/RP_{max} \rceil + 1 + m \cdot LP_i,$$

where $\lceil m/ RP_{max} \rceil + 1$ is the number of ACKs sent by receiver node i, including the ACK for the last packet of the transmission and $m \cdot LP_i$ is the number of NAKs sent by the same node. Hence the total number F_{RAND} of feedbacks received by the repair node from its N receiver nodes will obey the inequality

$$F_{RAND} \leq \sum_{i=1}^{N} (\lceil m/RP_{max} \rceil + 1 + m \cdot LP_i) \quad (3)$$

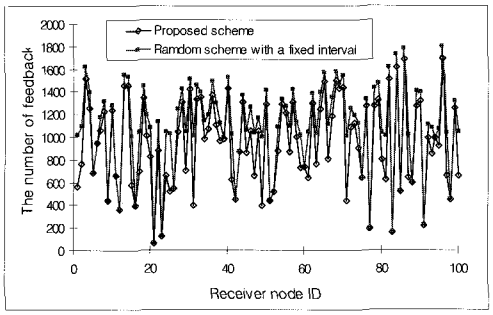


Fig. 2. The number of feedbacks from both schemes.

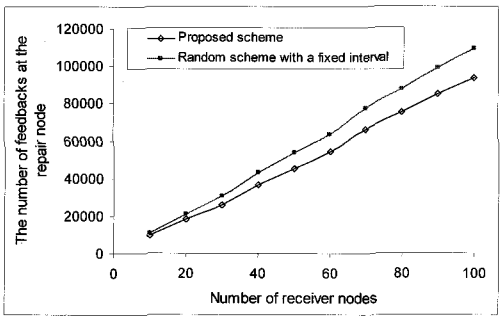


Fig. 3. The number of feedbacks sent by different number of receiver nodes.

On the other hand, the number of feedbacks F_{ACK} for a pure ACK-based scheme, where all receiver nodes acknowledge all the packets they receive, is given by

$$F_{ACK} = mN$$

Under the same assumptions, proposed scheme can reduce the number of feedbacks because it allows the receiver nodes to adaptively send the feedbacks depending on their packet loss patterns. That is, they slightly increase their intervals if there is no packet loss during their previous intervals and dramatically decrease it otherwise. Since today's link is fairly reliable, they will send their feedbacks with increased intervals in many cases.

Fig. 2 shows the number of feedbacks for the two schemes. Note that the packet losses within the range of spatial locality among the group are not counted because they will not generate any feedbacks from the receiver nodes owing to the *NAK suppression* message sent by repair node. In this experiment, we set the m to 10,000, which roughly represents a transfer of

10 megabytes when the packet size is 1 kilobyte. We also set the RP_{max} to 10.

We assumed that the individual loss probabilities LP_i would be uniformly distributed between 0 and 0.1. As we can see, the proposed scheme can reduce more than 15% of feedbacks from the receiver nodes compared to the randomized scheme with a fixed interval.

Fig. 3 shows how the number of feedbacks at the repair node is affected by the number of receiver nodes per repair node. This result indicates that our scheme provides efficient buffer management functionality for repair node by reducing the number of feedbacks sent by receiver nodes. As we can see, the performance gap between both schemes is more and more prominent as the group size increases. This feature provides scalability since each repair node will be able to handle more receiver nodes. We also need to mention that minimum difference between our scheme and the pure ACK-based scheme is about one million feedbacks.

4.2 Buffer Space

The sole disadvantage of our scheme is that each repair node will use more buffer space than a repair node receiving ACKs from all its receiver nodes for all packets sent by the sender node.

The total buffer space used by all repair nodes will however remain almost constant because repair nodes receiving less ACKs will be able to manage more receiver nodes. As a result, we will have fewer repair nodes with larger buffers. To show that let us assume that a repair node can receive up to n_{ACK} ACKs for each packet sent by the sender node. This means that a repair node receiving ACKs for all packets sent by the sender node will be able to handle up to n_{ACK} receiver nodes and will need a buffer capable of storing one single packet. Under our scheme, each packet is acknowledged by much less than $1/RP_{max}$ of the receiver nodes in average. This means that each repair node will be able to handle much more than $n_{ACK} \times RP_{max}$ receiver nodes. The maximum buffer space requirements of each repair node will be multiplied by RP_{max} but the total buffer space requirements of all repair nodes will remain unchanged because we will need about RP_{max} less of them under the worst assumption.

4.3 Error Recovery Delay

The additional retransmissions increase the error recovery delay because the repair node cannot retransmit the requested packet immediately. This will happen when the repair node does not have the requested packet in its buffer. The packets should be retransmitted from its original sender node or upper stream repair node. This might double or triple the error recovery delay. Also, these additional retransmissions cause unnecessary traffics between the repair nodes in the different levels of the logical error recovery tree.

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 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.

$$P(N_A(\delta) \geq B) = 1 - \sum_{n=0}^{B-1} \frac{(\lambda\delta)^n e^{-\lambda\delta}}{n!} = 0, \quad (4)$$

which simplifies into $\sum_{n=0}^{B-1} \frac{(\lambda\delta)^n}{n!} = e^{\lambda\delta}$. Since we have $e^{\lambda\delta} = \sum_{n=0}^{\infty} \frac{(\lambda\delta)^n}{n!}$, (4) 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 [10], 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's buffer while their retransmission could still be requested by one or some of the receiver nodes. In this case, the missing packets will have to be resent from either an upper repair nodes or the sender node. In most cases, the packets will have to be resent 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 traffics decreasing the whole Internet performance.

The proposed scheme requires extremely a few additional retransmissions either from its upper-stream repair nodes or sender node because the repair node almost always has in its buffer all packets that can be requested by any of its receiver nodes. This feature provides fast error recovery for receiver nodes because it only requires one round trip time between the receiver and repair node plus a packet processing time at the repair node. The only case where the additional retransmissions are needed is when all of the receiver nodes experience a packet loss for the same packet and the repair node did not correctly receive the packet. But, this is theoretically a rare event with extremely low probability because this is an independent event among the group. It will not be happened in a real environment with reasonable amount of data transmission.

V. CONCLUSION

We have proposed a feedback scheduling scheme combining NAKs and infrequent ACKs to provide scalability and reliability in a multicast session. Under our scheme, receiver nodes send NAKs to repair nodes to request packet retransmissions only when the packet losses are occurred by independent link errors. Otherwise, they refrain from sending them by the order of *NAK suppression* message sent by their repair nodes. At dynamic and infrequent intervals, they also send ACKs to their repair nodes to indicate which packets they can safely discard. The proposed scheme reduces delay in error recovery because the packets requested from the receiver nodes are almost always available in the repair node's buffer. It achieves this without increasing the server workload because (a) each receiver node only sends infrequent ACKs, and (b) their sending times are distributed among all the receiver nodes. In addition, it greatly reduces the number of repair nodes required to handle a given number of receiver nodes. Hence, it provides an

acceptable trade-off between ACK-based and NAK-based schemes to achieve reliability and scalability.

REFERENCES

[1] B. Adamson, C. Bormann, M. Handley, and J. Macker: NACK-Oriented Reliable Multicast Protocol (NORM). IETF draft-ietf-rmt-pi-norm-10, (2004)

[2] J. Baek and E. Lee: An Improved Logical Tree Construction Scheme for Tree-Based Reliable Multicast. Proc. of the ICTS, (2003) 110–121

[3] J. Baek and J. F. Pâris: A Buffer Management Scheme for Tree-Based Reliable Multicast Using Infrequent Acknowledgments. Proc. of the IPCCC, (2004) 13–20

[4] J. Baek and J. F. Pâris: A Heuristic Buffer Management and Retransmission Control Scheme for Tree-Based Reliable Multicast. ETRI Journal, Vol. 27. No. 1. (2005) 1–12

[5] J. Baek and J. F. Pâris: A Tree-Based Reliable Multicast Scheme Exploiting the Temporal Locality of Transmission Errors. Proc. Of the IPCCC, (2005) 275–282

[6] J. Baek and M. Kanampiu: A NAK Suppression Scheme for Group Communications Considering the Spatial Locality of Packet Losses. Int. J. Computer Science and Network Security, Vol. 6. No. 10. (2006) 158-167

[7] M. Costello and S. McCanne: Search Party: Using Randomcast for Reliable Multicast with Local Recovery. Proc. of the IEEE ICC, (1999) 1256–1264

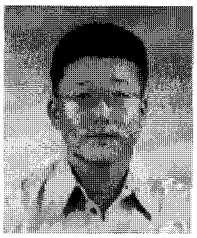
[8] S. Floyd et al: A Reliable Multicast Framework for Lightweight Sessions and Application-Level Framing. IEEE/ACM Transactions on Networking, Vol. 5. No. 6. (1997) 784–803

[9] M. Kadansky et al.: Reliable Multicast Transport Building Block: Tree Auto-Configuration. IETF draft-ietf-rmt-bb-tree-config-01.txt (2000)

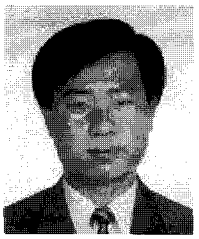
[10] S. K. Kasera, J. Kurose, and D. Towsley: Buffer Requirements and Replacement Polices for Multicast Repair Service. Proc of the NGC, (2000) 5-14

[11] S. J. Koh et al.: Configuration of ACK Trees for Multicast Transport Protocols. ETRI Journal, Vol. 23. NO. 3. (2001) 111–120

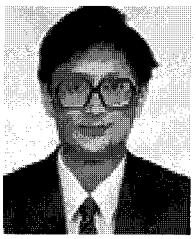
[12] J. C. Lin and S. Paul: RMPT: A Reliable Multicast



Jinsuk Baek is Assistant Professor of Computer Science at the Winston-Salem State University (WSSU). He is the director of Network Protocols Group at the WSSU. He received his B.S. and M.S. degrees in Computer Science and Engineering from Hankuk University of Foreign Studies (HUFS) in Yougin, Korea in 1996 and 1998, respectively and his Ph.D. in Computer Science from the University of Houston in 2004. Dr. Baek was a post doctorate research associate of the Distributed Multimedia Research Group at the University of Houston. His research interests include scalable reliable multicast protocols, mobile computing, network security protocols, proxy caching systems, and formal verification of communication protocols. He is a member of the IEEE.



Cheonshik Kim received his B.S. degree in Computer Engineering from Anyang University, Korea, 1995, and M.S. degree in Information Engineering from Hankuk University of Foreign Studies (HUFS), Korea, 1997 and Ph D. degree in Computer Engineering from HUFS in 2003. He joined the faculty of Anyang University, Korea where he is currently a professor in Department of Digital Media Engineering. His research interests include Multimedia systems, Distance learning, Databases, Data mining, and Information Security. He is a member of Institute of Webcasting, Internet TV, and Telecommunication, Korea Society for Internet, Society of Transportation.



You-Sik Hong was born in Seoul, Korea, in 1959. He received the B.S. degree in Electronic Engineering from the KyungHee University, Seoul, Korea, in 1983, the M.S. degree in Electronic Engineering from New York Institute of Technology, New York, U.S.A., in 1983, and the Ph.D. degree in Electronic Engineering from the KyungHee University, Seoul, Korea, in 1997. From 1989-1990, he was a research engineer at the Samsung company in Korea. He is currently Full professor in the Department of Computer Science, SangJi University, Wonju, Korea. His research interests are in the areas of fuzzy rule, expert system, fuzzy expert system, and neural network problems.