A Study of Proactive Hybrid FEC/ARQ and Scalable Feedback Techniques for Reliable, Real-Time Multicast

Dan Rubenstein, Jim Kurose, and Don Towsley

Computer Science Department
University of Massachusetts Amherst
{*drubenst, kurose, towsley*}*@cs.umass.edu*

Abstract

In this paper, we examine the performance benefits of using parity encoding for reliable delivery of data to multiple receivers, where the receivers have heterogeneous loss and delay characteristics. First, we examine the effect of sending parity loss repairs *proactively* to receivers before it is known whether or not the repairs are needed to repair losses. We show that in many cases, proactive repair can reduce feedback implosion and the expected delay of reliable delivery without increasing bandwidth usage. Next, we examine how proactive repairs can be used to meet end-to-end deadlines to within a tunable probability. Last, we examine several receiver feedback mechanisms for parity repairs in a heterogeneous networking environment, and find that the various mechanisms offer different levels of robustness. Our examinations take into account temporally correlated (bursty) loss, as well as loss of feedback messages from receivers.

Key words: Forward Error Correction (FEC), Reliable Multicast, Real-time delivery

1 Introduction

The Internet enables the transmission of a variety of types of data. In particular, there is a growing need to reliably transmit real-time data to numerous receivers where data must be received within a bounded amount of time. Examples range from video and audio communication to reliable delivery of stock data. Using multicast instead of multiple point-to-point unicast connections

to distribute this information reduces network bandwidth required to support such applications. However, dealing with packet loss for time-constrained multicast applications remains an open and challenging research problem.

There are several issues that reliable multicast protocols must address. First, protocols must avoid *feedback implosion*: an aggregation of feedback from numerous receivers to the sender. For instance, it is well understood that protocols that require each of a large number of receivers to notify the sender of a packet loss (NAK) will overwhelm the networking resources close to or at the sender [TKP97]. It is also important to reduce the bandwidth used for retransmissions. Previously proposed solutions solve this problem by introducing delay [TKP97], or by distributing the repair process among the session participants in order to reduce the number of requests that must be sent to the sender [FJL+97,PSLS97,KKT98,PPV98,SBE+98,LDW97]. These works are targeted toward applications that do not have tight time constraints. The unpredictability of the location of the repairing participant complicates the timing and coordination of retransmissions that is required for real-time delivery where the deadline is fixed. The STORM protocol [XMZ97] is designed for real-time delivery, in that it ceases requesting repairs after a deadline has passed. However, again due to the unpredictability of the location of the repairing participant, a receiver cannot adjust the level of reliability with which it receives packets, except by modifying the deadline itself.

In this paper, we focus on the use of parity encoding techniques to reduce the latency of reliable delivery of data to multiple receivers and to reduce feedback implosion. Such techniques are useful in applications that deliver real-time information to a large receiver set and can tolerate small transmission delays. These applications include high quality, non-interactive audio and video, and stock quote dissemination. First, we examine an approach that sends parity-encoded loss repairs *proactively*, i.e., repairs are sent before it is known whether the transmission of the repair is necessary. If these proactive repairs are insufficient to reliably deliver the data to receivers, additional FEC repairs are obtained via receiver requests for the transmission (ARQ) of additional repairs. We find that the bandwidth used by the sender for data and repair transmission does not increase significantly in comparison to that used by a non-proactive protocol, as long as the number of packets transmitted proactively is kept below a certain bound. At the same time, there is a significant reduction in both the expected delay of reliable delivery, as well as the amount of feedback transmitted toward the sender.

Next, we look at how a receiver, with knowledge of its end-to-end loss rate and round-trip-time from the data source, can proactively request repairs in order to meet a fixed deadline for reliable delivery to within a high, tunable probability. We describe the computations that must be performed by the receiver to determine the number of repairs that it should request in order to

meet such a deadline, taking into account the fact that packet losses occur in bursts. We determine the bandwidth overhead that results from requesting these additional repairs, and demonstrate that the bandwidth overhead can be decreased by having the sender send repairs proactively along with its initial transmission of data.

Last, we propose and evaluate several feedback mechanisms from the receivers to the sender that differ in the information that is returned to the sender. The assumption is made that receivers have different round trip times to the sender, and feedback requests can be lost. We consider several kinds of feedback, the largest sequence number received, the number of packets still needed, and/or a counter that suppresses certain redundant requests. We find little difference among them except for one version that maintains a separate counter for each round that tracks the number of repairs sent for that round. Surprisingly, such feedback causes the sender to transmit a significant number of additional, unnecessary repairs, due to a lack of coordination of the requests from the various receivers over the various rounds. Last, we briefly look at how these techniques can be combined into a single protocol for use in large-scale multicast applications requiring low latency or having real-time deadline requirements.

In order to focus on the performance implications of proactive FEC, we make several simplifying assumptions about the network model and the protocol. We assume that link loss rates are not affected by the rate at which the protocol transmits. This is reasonable in a scenario where the congested links utilized by the protocol are also utilized by many other sessions. We model link loss as a two-state Markov process, which has been shown to be an effective means for modeling bursty loss [YMKT99]. We do not consider hybrid techniques that combine proactive FEC transmission with local recovery techniques [Ker98]. Our work in [RKTK99] demonstrates that in certain multicast session configurations, a hybrid protocol can further reduce bandwidth requirements of reliable multicast protocols. In contrast, Nonnenmacher et al, [NLJ+98], demonstrate that in many scenarios, there is no additional benefit to adding repair servers. It has been argued that specially designed routing algorithms that provide repair functionality may obviate the need for FEC techniques [LC99]. These algorithms also require routing changes which have yet to be implemented. Furthermore, our approach to bounding the delay on reliable delivery requires conservative estimates of round trip times to repairing entities. This estimation is complicated when repairs originate from multiple network entities.

The rest of the paper proceeds as follows. Section 2 presents a simple overview of the use of parity encoding in reliable multicast. In Section 3, we examine two methods by which proactive transmission of repair packets can be used to improve protocol performance. In Section 4, we compare the performance of

several feedback mechanisms that can be used by receivers in a heterogeneous environment. We briefly describe how these ideas can be combined to produce a scalable real-time reliable multicast protocol for heterogeneous environments in Section 5. We discuss previous and related work in Section 6, and summarize the paper in Section 7.

2 FEC primer

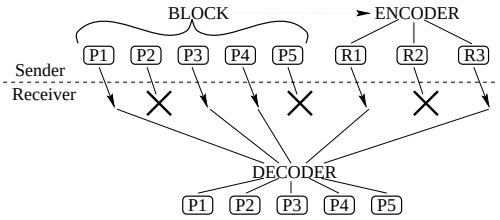


Fig. 1. A sample decoding using a block built from 5 packets.

Error correcting codes were initially developed to correct erroneous or missing bits. More recently, they have been applied to repair packet losses at the network layer. The following description is sufficient for understanding how systems equipped with Reed-Solomon encoders and decoders [Bla83] can make use of repair packets to recover from loss.

The sender forms *blocks* containing k data packets, where each block consists of a subset of the data packets it wishes to deliver reliably. We refer to k as the *block size*. The sender inputs a block into its encoder which then generates *repair packets* for that block. A receiver uses its decoder to recover the k data packets from any combination of k distinct packets consisting of a mix of the block's original data packets and repair packets generated for the block. We say that the set of data packets within a block and repair packets generated from those data packets *belong* to the block. When a receiver obtains k packets (either data or FEC repair packets) for a block of size k, we say that the receiver has completed receipt of the block.

An example is shown in Figure 1, where a sender forms a block of 5 data packets, encodes 3 repair packets from this block, and transmits all 8 packets to the receiver. As soon as the receiver receives any 5 distinct packets related to the block (in the example, 3 data and 2 repair), it activates the decoder and recovers the lost data packets.

Detailed discussions of Reed-Solomon coding techniques and of packet-level FEC techniques can be found in [McA90] and [NBT98] respectively; implementation issues are considered in [Riz97,NBT98]. For our purpose here, we simply note that FEC techniques exist that can be used to generate as many repair packets as needed and that, as of 1997, this could be done at data rates

much higher than 8 Mbytes/sec on commodity PC's.

3 A Proactive FEC+ARQ Technique

In this section, we look at protocols that use a combination of ARQ and FEC to reliably deliver data to a set of receivers, $\mathcal{R}$ [DL95,NBT98]. The novel aspect of these protocols is the use of techniques that proactively transmit FEC repair packets. To simplify our discussion, we restrict our consideration to protocols in which only the sender transmits repairs, and these repairs are always multicast. Furthermore, in this section, we assume that the receivers are homogeneous, i.e., the round-trip time from the sender to each receiver is fixed and identical across receivers. Hence, feedback from receivers in response to a given transmission from the sender reaches the sender at the same time, and repairs and data packets that are multicast by the sender reach receivers at the same time. We can therefore envision the communication between the senders and the set of receivers taking place over a series of *rounds*. In Section 4, we will resolve issues that arise when the assumption that round-trip times to the sender vary across the set of receivers.

We focus here on a single block of data and determine the number of rounds that are necessary to reliably deliver the data to all receivers. In the initial round, the sender multicasts packets (the k data packets and perhaps some FEC repair packets). Any receivers that fail to receive at least k packets notify the sender. If the sender receives any feedback, a subsequent round commences with the sender transmitting a set of repair packets. This process continues until all receivers have received at least k packets.

We now describe the basic protocol used by the sender and the receivers to transmit a block of k data packets. The sender multicasts the k data packets during the first round, with zero, one, or more FEC repair packets (what determines this number is discussed below). We say that any repairs transmitted at this time are transmitted *proactively*, because it is not known whether or not receivers will use them. Due to potential loss inside the network, each receiver receives a subset of these transmitted packets. The subsets received by the various receivers need not be identical since a multicast packet may be lost at various points within the network. After the sender's first round transmission ends, each receiver assesses the total number of distinct packets that it has received. If receiver i has received $k_i < k$ packets belonging to the block, it sends a request to the sender to transmit FEC repair packets. Henceforth, we will refer to this request as a NAK, since such a request acknowledges that an insufficient number of packets was received. For now, we assume that a NAK specifies only the number of additional FEC repair packets desired by that receiver for that block: the use of FEC obviates the need for the sender

to know specifically what packets have been received by the receiver. By requesting more than $k - k_i$ additional repairs, a receiver is requesting repairs proactively. In the next round, the sender transmits $\max_{i \in \mathcal{R}} k_i$ additional repairs. This process repeats until the receivers have obtained the k packets. An entire stream of data is transmitted by segmenting the data packets into blocks, and applying the above steps to each block.

The reduction in bandwidth resulting from the use of FEC-based repairs is analyzed in depth in [NBT98]. Here, we will show how changing the level of proactivity affects several attributes of a reliable transmission. These attributes are:

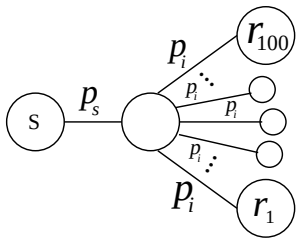
- **Expected Delay:** We assume that the delay in reliably transmitting a block to a receiver is proportional to the number of rounds that it takes the receiver to obtain at least k packets for that block, where k is the block size. For non-real-time protocols, we will be interested in the expected delay to reliably deliver a block.
- **Deadline Failure Probability:** For a protocol that imposes a deadline on reliable data delivery, we will be interested in the probability that a block is received reliably before a given deadline.
- **Implosion factor:** The expected number of NAKs that a sender receives per block.
- **Forward bandwidth:** The expected number of packets (repair and data) sent by the sender per block. Because the sender multicasts each packet it sends and each packet traverses all links in the multicast tree (except links on the tree that lie below points of loss of a packet), the bandwidth usage in the network is approximately a linear function of the forward bandwidth.

We consider two styles of proactive FEC. The first style is sender-oriented, in which the sender transmits repairs proactively in the initial round. We will observe that this style of proactivity can significantly reduce the implosion factor and expected delay while increasing forward bandwidth by only a negligible amount. The other style is receiver-oriented, in which the receiver requests a number of repairs that is larger than the minimum necessary to complete receipt of the block. We will observe that receiver-oriented proactivity is useful in meeting real-time deadlines with high probability. These two styles are orthogonal and can be easily combined. We now describe each of these styles separately and in greater detail.

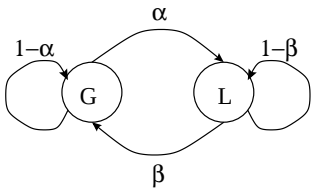
3.1 Sender-oriented proactivity

We associate a *proactivity factor* ρ with a sender-oriented proactive protocol. Here, ρ is a real number larger than or equal to one. For each block of size k,

the sender initially transmits $\lfloor \rho k \rceil$ packets,[1] consisting of the k data packets plus an additional $\lfloor (\rho - 1)k \rceil$ repair packets. At the end of each round, each receiver requiring additional packets for the block transmits a request for the number of additional repairs it needs in order to receive k packets.



(a) A modified star topology



(b) A two state loss model

Fig. 2. Network topology and loss model

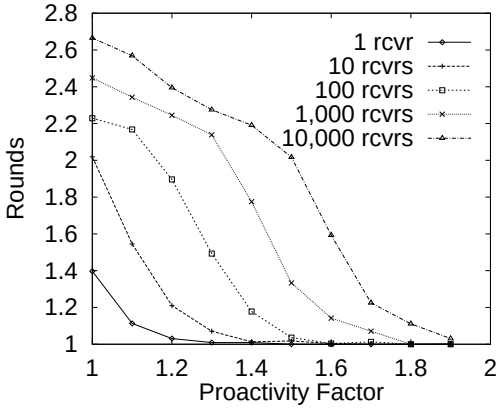
Let us now quantify the benefits of sender-oriented proactivity. We assume that the multicast tree is a modified star as illustrated in Figure 2(a). The sender transmits packets across a shared link that lies on the path to all receivers. If the packet not lost on the shared link, it is transmitted on each fanout link. Each transmitted packet is dropped on the shared link with probability p_s. If not dropped on the shared link, the transmission on each fanout link is dropped with probability p_i. We model losses on each link as a discrete time two-state loss process (Figure 2(b)). This allows us to easily model bursty losses as they occur on the Internet [YMKT99].

A state transition occurs each time a packet arrives at a link. If the process transits to state G, then the packet passes through the link. Otherwise, the process transits to state L and the packet is lost. Given that the process resides in state G when a packet arrives, it transits to state L with probability α, and remains in state G with probability $1 - \alpha$. Given that the process resides in state L when a packet arrives, it transitions to state G with probability β, and remains in state L with probability $1 - \beta$.

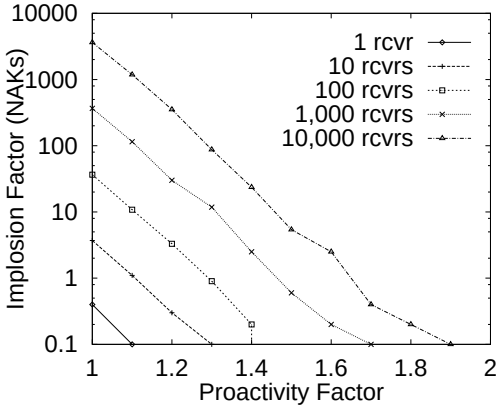
We use two parameters to specify a link's loss process, the loss rate and the burstiness. Let p denote the probability that a random packet is lost. Let B

[1] The notation $\lfloor x \rceil$ means round x to the nearest integer.

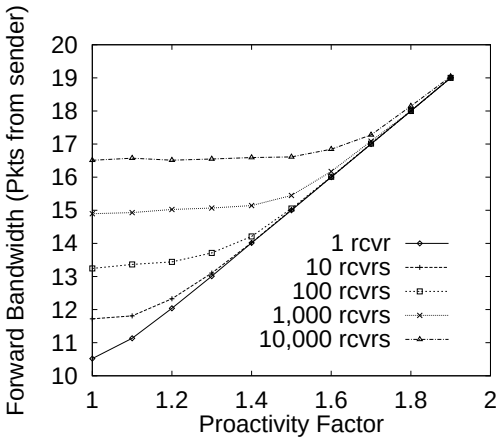
denote the *burstiness* if the expected number of packets lost in a row on the link is $B/(1-p)$. Note that a Bernoulli process has a burstiness of $B = 1$.



(a) Rounds



(b) NAKs



(c) Forward Bandwidth

Fig. 3. Sender-oriented proactivity results

The results in [YMKT99] indicate that typical values of B in the Internet are less than 1.2. We will use $B = 1.2$ in the results presented here. Hence they are expected to give pessimistic results regarding the gains in performance due to proactive FEC.

Figures 3(a), 3(b), and 3(c) plot simulation results from 10,000 experiments. These figures respectively plot the expected number of rounds (our measure of expected delay for round-based protocols), implosion factor, and forward bandwidth per block for various multicast group-sizes as a function of the sender's proactivity factor. The block size is 10, and the loss probability on each link is .02532 (roughly an end-to-end loss probability of .05). From Figure 3(a), we see that a protocol using a proactivity factor slightly larger than one requires a significantly smaller number of rounds than one that uses a proactivity factor of one. The expected number of rounds quickly converges to one as the proactivity factor is increased. In Figure 3(b), we see that the implosion factor decreases exponentially as a function of the proactivity factor, so that it can be significantly reduced by a small increase in the proactivity factor. The most interesting result is the effect that the proactivity factor has on forward bandwidth as shown in Figure 3(c). For large multicast groups, initially increasing the proactivity factor has an insignificant effect on the expected number of packets transmitted to reliably deliver data to all receivers. This is because, with a large group, a certain number of repairs must be expected. With proactive FEC, these repairs are simply sent before it is known for certain that they are needed. Note that, after some point, an increase in the proactivity causes the sender to transmit more packets than are required by any of the receivers, and the bandwidth begins to increase along the asymptote $y = kx$, where k is the block size.

Thus, one can obtain a significant decrease in delay (as shown in Figure 3(a)) without incurring any significant increase in forward bandwidth. Clearly, such a protocol's performance compares favorably in every respect to that of a non-FEC ARQ protocol without local recovery. For example, a non-FEC ARQ protocol delivering data to 10,000 receivers would require on average approximately 4 rounds to reliably deliver a block of 10 packets to all receivers, would transmit approximately 30 repairs, and would generate on the order of 6,500 NAKs.[2] In contrast, we observe from Figure 3 that a hybrid FEC-ARQ protocol using a proactivity factor of 1.6 requires on average approximately two rounds, 7 repairs, and 10 NAKs.

We emphasize that our results present expected values and transmitting re-

[2] The count of 6,500 NAKs assume that receivers do not suppress the transmission of NAKs. NAK suppression is commonly used in reliable multicast protocols to reduce implosion, but can add additional and unpredictable latencies that would complicate real-time reliable delivery.

pairs proactively does not lower the worst case bounds on implosion and delay. For instance, consider the case where packet loss occurs on a link near the sender that lies on the transmission path to a large set of receivers. While increasing the proactive factor decreases the likelihood that these receivers will transmit NAKs, when they do need additional repairs, they all transmit NAKs. For cases where there is significant spatial correlation of packet losses among the receivers, increasing the proactive factor decreases the frequency with which implosion occurs, but does not reduce the number of feedback messages when additional transmissions are required. Feedback suppression techniques that introduce randomized delays such as those discussed in [BTW94,NB99] can significantly reduce the expected number of receivers that transmit feedback even when packet losses are spatially correlated. However, these techniques increase the expected delay and also fail to lower the worst case bounds. Tree-based solutions that aggregate feedback from receivers as the feedback moves up the multicast tree can alleviate the feedback problem altogether (e.g., see [LPGLA98,LLGLA96]). However, these solutions impose additional processing and state at intermediate nodes within the tree. By combining a tree-based solution with proactive FEC techniques can reduce the expected amounts of processing and state that these intermediate nodes incur.

Extensive simulation over a variety of loss rates and various homogeneous and heterogeneous network topologies appears in [RKT98]. This extensive testing has shown that the network topology has little impact on performance, as long as the end-to-end loss rates and burstiness of the loss are fixed. In [RKT98], we also conclude that bursty loss can have a significant impact on the protocol's performance. The performance improvements achieved by using FEC are less evident as the burstiness increases. This is because an increase in burstiness increases the variance of the number of packets that require retransmission at the end of a round. This means that for any fixed proactivity factor, as the burstiness is increased, there is an increase in both the expected number of additional repairs that must be transmitted during successive rounds and in the expected number of repairs that are transmitted unnecessarily in the first round.

3.2 Receiver-oriented proactivity

We have seen how sender-oriented proactivity reduces the expected latency and feedback in reliable multicast delivery without significantly increasing the forward bandwidth from the sender. We now consider how proactive delivery can be applied at the receiver side to meet real-time guarantees. We consider an application in which there is a deadline, τ, by which a complete block of packets must be received by each receiver, and a lower bound, $\mathcal{P}$, on the

probability that the receiver receives the entire block without loss before the deadline.

We assume here that the deadline is specified as a number of rounds. The receiver must be able to calculate the probability of receiving n out of m packets, which we define as $Q(m, n)$. We now present a recursive construction for $Q(m, n)$: the probability that at least n of m packets are received given that the loss process is described by a two-state loss model as discussed previously.

Let $Q_G(m, n)$ to be the probability that the receiver successfully receives at least n of m transmitted packets, and that the last repair is received (i.e., the burst loss model ends in state G). Similarly, let $Q_L(m, n)$ to be the probability that the receiver successfully receives at least n of m transmitted packets, and that the last packet transmitted is not received (i.e., the burst loss model ends in state L). We further assume that the probability of being in a given state at the start of the transmission equals the steady state probability for that state. In effect, this states that the time between sender transmissions of separate feedback messages is large enough to remove any temporal loss correlation between the transmissions. Define π_G and π_L to be the steady state probabilities of being in states G and L, respectively. Then

$$\pi_G = \beta/(\alpha+\beta) \quad \pi_L = \alpha/(\alpha+\beta)$$

$$Q_L(m,n) = \begin{cases} 0, & m = n > 0 \\ \pi_L, & n = 0 \\ Q_L(m-1,n)(1-\beta) + Q_G(m-1,n)\alpha, & m > n > 0 \end{cases}$$

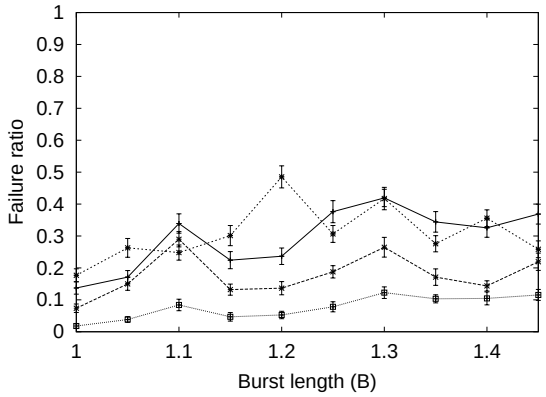
$$Q_G(m,n) = \begin{cases} \pi_G(1-\alpha)^{n-1}, & m = n > 0 \\ \pi_G, & n = 0 \\ Q_L(m-1,n-1)\beta + Q_G(m-1,n-1)(1-\alpha), & m > n > 0 \end{cases}$$

$$Q(m,n) = Q_L(m,n) + Q_G(m,n)$$

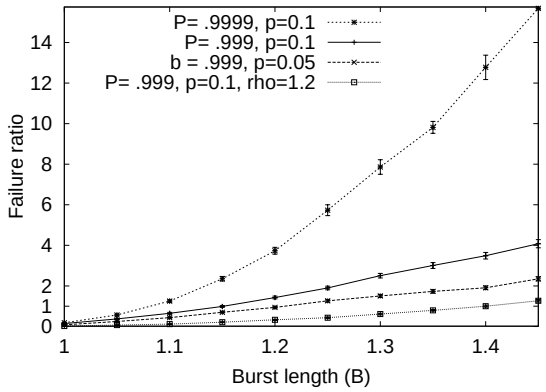
3.2.1 Evaluation

We begin by evaluating the use of $Q(m, n)$ to meet a deadline in an environment where losses are bursty. We define the *failure ratio*, $F_r = p_f/(1-\mathcal{P})$, where p_f is the probability that a receiver fails to recover a block of data within the deadline. A failure ratio less than or equal to one indicates that the receiver fails to meet deadlines with probability less than $\mathcal{P}$.

Figure 4(a) shows that using $Q(m, n)$ to meet deadlines keeps the failure ratio



(a) Using $Q(m, n)$



(b) Using $Q_B(m, n)$

Fig. 4. The degradation of $Q(m, n)$ and of $Q_B(m, n)$ as loss becomes bursty

well below 1, even as the burstiness (B) of the loss process is increased. We vary the normalized burst length, B, on the x-axis, and give the resulting failure ratio on the y-axis. The different curves represent various combinations of loss rate, p, and receiver deadline probability, $\mathcal{P}$. All curves have $\rho = 1$, except for the bottom curve, for which ρ is 1.2. For simplicity, we consider a multicast network with a single receiver. Point estimates and 95% confidence intervals are generated from 30 experiments, each consisting of $2/(1 - \mathcal{P})$ rounds of simulation. The number of rounds per sample point is inversely proportional to the bound on the probability that the deadline fails so that most samples contain a small number of failed attempts at meeting the deadline.

The failure ratio, F_r, lies below 1 because the minimum value of m is chosen such that $Q(m, n) \geq \mathcal{P}$. Hence, the actual probability that the block of data is received before the deadline is often larger than $\mathcal{P}$. We verified that $Q(m, n)$ was the minimum acceptable value to meet the deadline within the specified probability by simulating receiver-oriented proactivity where the receiver re-

quests $Q(m,n)-1$ packets in the last round (instead of requesting $Q(m,n)$). We found the failure ratio in this case to often be larger than 1.

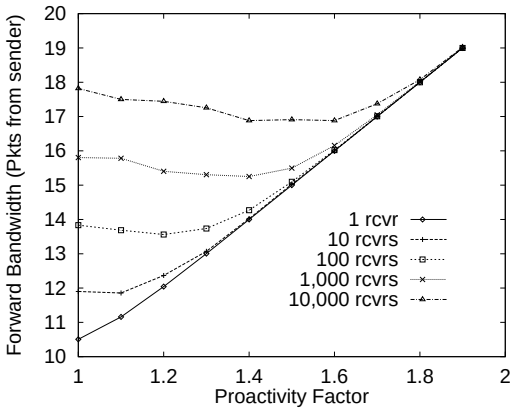
Let us consider what happens if a receiver does not take into account the burstiness of the loss process. Define $Q_B(m,n)$ to be the probability that at least m of n packets are received given that packet loss is described by a Bernoulli loss process with rate p. It is easy to verify that

$$Q_B(m,n) = \sum_{i=n}^{m} \binom{m}{i} (1-p)^i p^{(m-i)}. \tag{1}$$

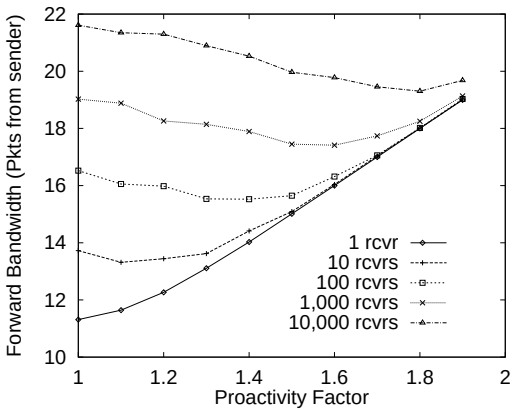
Figure 4(b) shows the failure ratio as the burstiness of loss increases when a receiver assumes that losses are described by a Bernoulli loss process. In this case, the receiver uses $Q_B(m,n)$ instead of $Q(m,n)$ to compute the probability of receiving m of n packets. We observe that, by using $Q_B(m,n)$, the receiver is less likely to meet deadlines as the burstiness, B, of the loss process increases. The failure ratio increases in a bursty loss environment as $\mathcal{P}$ increases, and also increases slightly when the end-to-end loss rate increases. We find that the addition of sender-proactivity decreases the failure ratio. This is because adding sender-proactivity decreases the likelihood that additional packet transfers will be necessary in subsequent rounds. Also, having additional receivers increases the likelihood of a given receiver meeting a deadline, since the inclusion of other receivers only increases the aggregate number of packets transmitted by the end of a given round.

Figures 5(a) and 5(b) show the expected number of transmissions by the sender in a network where the normalized burst length, B, equals 1.2, and the block size is 10, for respective end-to-end loss rates of .05 and .1. Each receiver requires that it receives a block of packets (so that it can perform decoding) by the end of the second transmission round with probability 0.999. We vary the proactivity factor of the sender on the x-axis. The various curves correspond to different numbers of receivers.

We observe that when receivers attempt to meet hard deadlines, the sender can reduce the expected number of transmissions of data and repairs per block by increasing the proactivity factor. This is because, in the absence of proactivity, the number of packets requested by a receiver to meet its deadline is significant whenever a second round is required. By sending a few extra packets in the first round, the sender often obviates the need for the second round altogether. This reduction of forward bandwidth due to sender proactivity is less pronounced as the end-to-end loss rate is decreased, as is shown by contrasting the slopes of the curves in Figure 5(b) with those in Figure 5(a). The reduction in forward bandwidth due to the use of receiver-oriented proactivity also becomes less significant as the receiver's deadline increases (i.e., as more rounds of packet



(a) e2e loss rate .05



(b) e2e loss rate .1

Fig. 5. Bandwidth consumption meeting deadlines in a bursty-loss environment

transmissions are permitted). As the deadline increases, the expected number of transmissions decreases, converging to the expected number of transmissions when there is no deadline.

The guarantee we present here ensures that the conditional probability that a block is received, given that the block has not been received by the last round, is larger than $\mathcal{P}$. It may be of more interest to simply ensure that the probability that a block is received over all rounds is larger than $\mathcal{P}$. In [RKT98], we refer to the latter as a block goodput guarantee, and present a method for reducing the number of packets to meet this type of guarantee. Further studies have shown that using a goodput guarantee rarely reduces the number of packets that the receiver requests in its final round. The reduction in the number of packets requested is significant only when loss rates are very high or $\mathcal{P}$ is very close to 1 (e.g. within 10^{-6}).

We conclude our discussion on receiver-oriented proactivity by pointing out

that the implosion factor can only decrease as receivers increase the number of repairs requested proactively. A receiver transmits a NAK for a block in the first round of the block's transmission only if it fails to receive at least k packets for that block in the first round. Since the number of packets transmitted in the first round is unaffected by the "level" of receiver-oriented proactivity, this "level" does not effect the number of receivers that transmit NAKs in the first round. In each subsequent round, a higher "level" of receiver-oriented proactivity results in an increase in the number of packets transmitted by the sender for that round. This in turn increases the likelihood that a receiver will have received at least k packets by the end of that round. The decrease in the number of receivers that need additional packets results in a decrease in the number of receivers that request additional repairs.

3.3 Summary

In this section, we have analyzed the benefits of using proactive repair in round-based protocols. By using proactive FEC at the sender, we can reliably deliver data sooner than is possible using traditional ARQ and with considerably less bandwidth. Furthermore, proactivity provides large decreases in repair bandwidth and reduces latency in reliable delivery of data when compared to a non-proactive FEC-ARQ hybrid approach. The proactive approach achieves these gains without using significantly more forward bandwidth than its non-proactive counterpart. By using proactive FEC at the receiver, receivers can meet hard deadlines up to a given probability. When receivers are attempting to meet deadlines, adding proactivity at the sender can reduce the expected number of packets that the sender transmits. The optimal level of proactivity for a session, both at the sender and at the receivers, is a function of loss rate. It is known that the loss rate will vary over time [YKT96,YMKT99]. Adjusting the level of proactivity over time such that it remains optimal for the network loss rate at each particular time remains an open issue.

4 Handling Heterogeneity in the network

In Section 3, we demonstrated the effectiveness of proactivity when coupled with a round-based protocol. Because the protocol is round-based, it adapts its delivery rate to the requirements of the most distant receiver (e.g., the receiver with the largest round trip time), reducing the protocol's effectiveness for closer receivers. We now examine how to eliminate this synchronous behavior. We describe and compare several feedback mechanisms that allow each receiver to communicate with the sender at a rate that does not depend on the number or locations of other receivers.

We first formalize a packet sequencing format used by the sender. Each packet (data or repair) is assigned a *sequence pair*, (i, j), where i is an integer indicating the block to which the packet belongs, and j is an integer indicating the packet's position within that block. If the ith block contains k packets, then data packets are assigned sequence pairs $(i, 0)$ through $(i, k-1)$, and FEC repairs for that block are assigned sequence pairs (i, j), where $j \geq k$. This mechanism for sequencing is commonly used in packet-level FEC protocols [NBT98]. The typical approach for requesting the number of additional repairs is as follows: at the end of each round, each receiver requiring additional repairs notifies the sender of the number of additional repairs that it requires. We assume that a technique, such as proactive FEC, has been employed to keep the implosion factor low so that the sender can effectively handle the requests for feedback that it receives. The sender collects this information from all receivers for that round and sends out a number of repairs equal to the maximum number requested. Such an approach works well when rounds are well defined. However, if receivers have differing round trip times and the sender does not delay retransmission to collect the information from all receivers, such an approach becomes considerably more complicated to coordinate.

We first describe the scalable feedback mechanism presented in [RKT98], which we refer to as *Largest Desired Sequence Number*, (or *LDSN*). Rather than requesting the number of additional repairs required, a receiver requests a packet sequence pair (i, j). The sender responds to any request (i, j) by transmitting all repairs (i, j') that have not been previously transmitted for all $j' \leq j$. Immediately prior to sending a feedback message, the receiver estimates the sequence pair (i, n) with the largest n that has been transmitted by the sender, and has or would have arrived at the receiver, barring loss. The receiver then returns $(i, n+m)$ (i.e., $j = n+m$), to the sender. Here m is the number of additional repairs desired by the receiver (the number of repairs required to complete the block plus any proactive quantity being requested). Note that some or all of the repairs $(i, n+1), \cdots, (i, n+m)$ are either already en-route to the receiver (perhaps due to some other receiver's request), or will be transmitted when the receiver's request reaches the sender. The burden of success or failure of this approach is moved to the receiver: it must correctly estimate the set of packets which have already arrived or are lost. Barring reordering, each receiver's estimate is recalibrated each time it receives a packet from the sender.

We consider three remaining approaches that make use of *NAK round counters* to coordinate repair transmissions from the sender. NAK counters are used loosely to maintain a round-by-round structure for repair transmission. Each NAK request from the receiver contains the number of repairs, n_r, that it requests for transmission, as well as a *NAK round*, N_r. The NAK round indicates the current "round" of the block transmission as perceived by the receiver. When the sender transmits repairs, it includes a NAK round number,

N_s, in its packet, corresponding to the largest NAK round it has received from any receiver for that block.

A receiver's NAK round counter, N_r, is initially 1. After sending a NAK, it increments its NAK counter ($N_r = N_r + 1$). Upon receiving a packet from the sender with a NAK round number larger than or equal to its own current value, the receiver updates its NAK round counter to the sender's current value plus one ($N_s \geq N_r \rightarrow N_r = N_s + 1$). By doing so, the receiver assigns a value to its NAK counter that it has not previously seen.

We consider three ways in which the sender can respond to a NAK requesting n_r repairs that has just arrived with round number N_r:

round-forward If $N_r \leq N_s$, then the sender ignores the NAK. Otherwise, set $N_s = N_r$ and transmit the n_r repairs requested by the NAK.

round-current The sender maintains a count, n, of the total number of repairs transmitted whose round number equals the current sender round number, N_s. For the arriving NAK, if $N_r < N_s$, then ignore the NAK. If $N_r = N_s$, transmit $\max(0, n_r - n)$ packets, and set $n = \max(n, n_r)$. If $N_r > N_s$, set $n = n_r, N_s = N_r$, and transmit n_r packets.

round-all For each integer $i > 0$, the sender maintains a count, n_i, of the aggregate number of packets transmitted by NAKs whose NAK round number was i. For the arriving NAK where $N_r = i$, let $n_{r'} = \sum_{j=i}^{N_s} n_j$. If $n_{r'} < n_r$, then transmit $n_r - n_{r'}$ repairs, and set n_i to $n_i + n_r - n_{r'}$. Otherwise, do nothing.

We say that repairs that are transmitted in response to a NAK with round number N_r are transmitted in the N_rth round. Round-forward only provides repairs in response to a NAK with a new, larger round number. As such, only one receiver can make a request per round. Round-current responds to new, larger round numbers in a manner similar to round-forward, but also transmits additional repairs in response to NAKs whose round number equals the largest round number seen previously. The number of repairs sent in a round equals the maximum number requested by receivers for that round. Round-all permits the transmission of repairs for round i after repairs have already been transmitted for round j where $i < j$. However, when sending repairs for round i, the number of packets transmitted is reduced by the number of packets that has been transmitted in any round $j \geq i$, since the receiver could not have possibly known about any of these transmissions, or it would have chosen a larger round number.

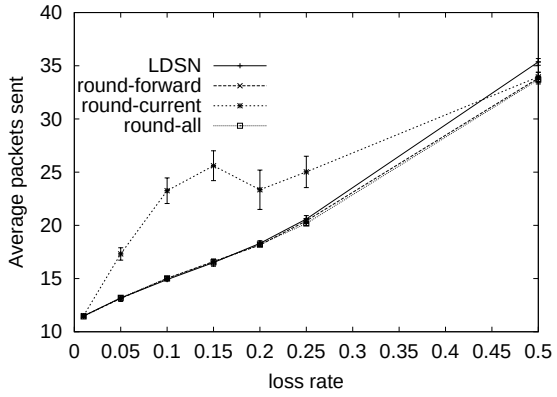
Let us discuss the pros and cons of the various approaches. We begin with the case that no NAK is lost. As long as ample time is given to allow repairs to reach the receiver that issues the repair request, LDSN and round-forward will transmit the minimum number of repairs necessary to repair all receiver

losses whereas one can easily construct scenarios where round-current and round-all can wind up transmitting unnecessary repairs. When NAKs can be lost, scenarios exist where LDSN and round forward will transmit unnecessary repairs.

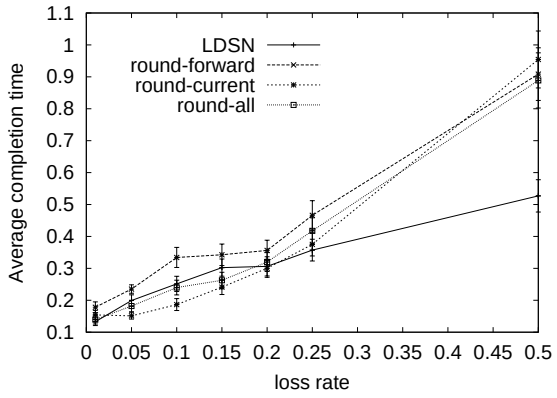
Figure 6 compares the four approaches in terms of the number of packets sent by the sender and the time it takes for all receivers to receive the block as a function of loss rate. measuring the number of packets sent by the sender. The sender continues to transmit packets as long as NAKs are received, and each receiver continues to send NAKs until it receives a number of packets equal to the block size. We also measure the time it takes for *all* receivers to receive a block of packets from the start of the initial transmission of the block of data. In these simulation experiments, losses are described by a Bernoulli process. There are 100 receivers, where the r-th receiver's round trip time is $.01r$. All packets transmitted (data, repair, NAK) are lost with equal probability, regardless of origin and destination. The block size is 10.

In Figure 6(a), we illustrate the average number of packets transmitted by the sender as a function of the packet loss rate. 95% confidence intervals are shown, where each sample point used to compute the confidence intervals is the average of 20 simulation runs. We see that the number of packets sent by the various approaches is roughly the same, except for the round-current approach, which, at loss rates less than .2 (i.e., at loss rates that are typical in the Internet), transmits more packets than the other approaches. At very high loss rates, LDSN transmits a few more packets than the other approaches. The inefficiency of the round-current approach can be understood through the following scenario: Let receiver A have a round trip time smaller than that of receiver B. Both receivers send NAKs with round number 1, with B requesting n_b repairs, and A requesting n_a repairs with $n_b > n_a$. Assume the sender gets the request for n_a repairs first, and multicasts out the repairs. If some of these repairs are lost on the path to A, then A will request an additional set of repairs with round number 2, prior to the arrival (or its knowledge) of the $n_b - n_a$ additional repairs being transmitted with round number 1. Hence, the repairs requested in round 2 are not all necessary, but because the round number in the NAK is larger, all packets are transmitted. We suspect that such a scenario can occur frequently in practice.

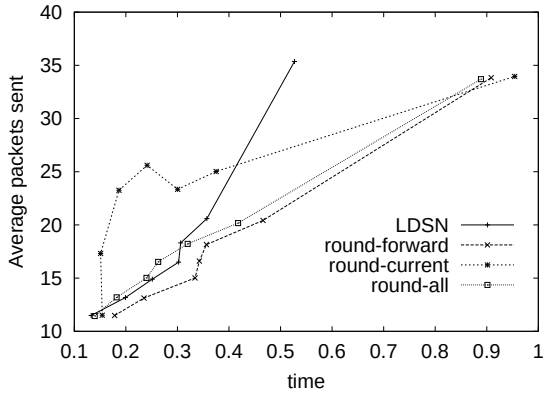
Figure 6(b) illustrates the completion time, the minimum time by which all receivers will be able to decode the block, as a function of loss rate. We see that for moderate loss rates, the round-forward approach takes the most time, due to its conservative nature of releasing additional packets. In contrast, the round-current and round-all approaches take less time due to their more optimistic approach at repairing losses. The LDSN approach maintains a low repair time even when loss rates are high. This is likely due to its lack of reliance on round numbers. When using round numbers, NAKs from a receiver



(a) Packets sent vs. loss rate



(b) Completion time vs. loss rate



(c) Packets sent vs. delivery time

Fig. 6. Bandwidth and Delay comparison of various forms of feedback.

with a small round trip time that experiences a single loss per round causes the sender to ignore the requests of a distant receiver (with large round trip

time) that loses a large set of packets in the initial transmission, and hence requests a large number of repairs in its NAK.

In Figure 6(c), we plot the completion time (x-axis) versus the average number of packets transmitted by the sender (y-axis). We omit the confidence intervals for clarity. We see that round-forward is the best approach to reduce the number of packets transmitted. However, if we also want to limit the completion time for moderate to high loss rates with only a small number of additional packet transmissions, LDSN is the best approach. It is interesting to also note that LDSN and round-forward both require the sender to only maintain one data value per block being transmitted: under round-forward, it must maintain the largest round number yet seen for the block, and under LDSN, it must maintain the largest sequence number transmitted so far. Our studies indicate that LDSN and round-forward are the two most effective means for receivers to send feedback to the sender.

5 Discussion

In this section, we briefly discuss how the techniques described in the previous sections can be combined to produce a real-time, reliable multicast protocol for heterogeneous networking environments. We have developed such a protocol. However, details of the protocol are omitted and can be found in [RKT98].

Our protocol operates on a block-by-block basis. Each block is handled separately by the protocol. The sender chooses a proactivity factor and transmits a proactive set of repairs along with the block of data. Doing so reduces the expected delay of receipt by receivers and also reduces the expected amount of implosion. The proactivity factor can be chosen such that the forward bandwidth from the sender (including subsequent retransmissions for the block) does not increase. The sender can adapt the proactivity factor to account for network conditions. If the sender is being heavily inundated with feedback, then it should increase the proactivity factor. If the sender is receiving little feedback, then it should reduce the proactivity factor.

Receivers solicit repairs from the sender using one of the heterogeneous feedback mechanisms discussed in Section 4. If receivers require the data before a given deadline, d, then they should develop conservative estimates of the loss rate and their round trip time, and use the techniques presented in Section 3 to meet these deadlines.

We briefly address several open issues not addressed in this paper, and present previous and ongoing works of other researchers that may solve these issues. First, we do not address the issue of NAK loss when a receiver attempts to

meet a hard deadline. No matter how many packets a receiver requests, its probability of receiving the block is compromised by the loss of the NAK. This impact of NAK loss can be reduced by transmitting the NAK several times. Note that the final NAK should be sent at a time that allows ample time for repairs to arrive prior to the deadline. However, there is often a gap between the transmission of the final NAK and the point in time where repairs have insufficient time to return. The receiver can minimize the possibility of NAK loss by periodically transmitting the NAK several times within this time interval. If the interval of time is not sufficient, then the receiver can begin transmitting its final NAK before the point in time we recommend. Note that the heterogeneous feedback mechanisms presented in this paper do not transmit additional repairs if the sender receives duplicate copies of a NAK.

We also do not directly address how one can apply these techniques to layered encoding schemes [MJV96]. We believe, however, that one can apply our proactive FEC techniques to each layer separately.

6 Related / Previous Work

In this section we discuss previous work that addresses various issues that arise when attempting to reliably multicast data in the Internet. Much of the work presented in this paper arises from observations that we made in [RKT98]. There is a significant body of previous work that discusses various mechanisms of providing reliable multicast in a scalable manner to a large numbers of receivers using randomly delayed transmissions [FJL+97], retransmission hierarchies [PSLS97,KKT98,PPV98,SBE+98], representatives [DO97], parity-encoding (FEC) [NBT98], and FEC-hierarchy combinations [Ker98,NLJ+98,RKTK99]. The primary concern of these works is to reduce bandwidth consumption and feedback implosion of a reliable multicast session. Often, the mechanism that provides this scalability also reduces the expected latency of reliable delivery. However, it is often difficult to determine a bound on the latency for each receiver, making it difficult to use the mechanism to enforce reliable delivery before a given deadline.

There has also been recent interest in providing *resilient multicast* service for real-time data, where retransmissions occur only if data can be delivered before the real-time deadline. Two protocols that are designed to provide resilient multicast are STORM [XMZ97] and LVMR [LPA99]. Both approaches form virtual trees with the source as the root and receivers as internal and leaf nodes. Recovery is implemented by sending all repair requests and retransmissions via unicast along this tree. Low latency repairs often can be performed provided that they do not need to traverse numerous links within the receiver-based tree. However, substantial delays occur when losses occur

close to the source. In such cases, the transmission path to a receiver can be significantly longer than the multicast route direct from the source. This is because repairs occur as a series of unicast transmissions between receivers. Hence, it is difficult to make time-bounded guarantees for reliable delivery.

Forward error correction (FEC) [Bla83] is a technique that reduces the bandwidth overhead of repairing errors or losses in bit streams. It has been shown in [NBT98] that a hybrid approach combining FEC with ARQ can significantly reduce bandwidth requirements of a large reliable multicast session. Hybrid approaches that combine FEC and ARQ have been proposed and classified for repairing loss and noise at the bit-level [DL95]. The Type I Hybrid approach involves sending repair bits within a packet that can correct bit errors or erasures, and retransmitting packets if the number of repair bits is insufficient to reliably receive the packet. Type II Hybrid approaches place the repair bits in packets that are distinct from the packets that contain the data. Our approach can be thought of as a Type II Hybrid approach. However, our forwarding of repair packets before their necessity incorporates certain favorable features of a Type I approach as well.

In [LDW97], real-time performance of reliable multicast techniques that only use FEC are compared to those only using ARQ techniques; [LDW97] does not consider hybrid approaches. An interesting approach using FEC is presented in [RV98], where data is delivered reliably without requiring ARQ through multicast group joins and leaves. A more recent implementation utilizes ARQ as a last resort to obtain any data that could not be obtained via the joining of a particular multicast group. A similar approach that uses layers to transmit a small number of proactive repairs is presented in [RJL+00]. However, present join-leave latencies for multicast groups make the approach too bandwidth inefficient to support real-time applications. Other works also show the promise of the proactive FEC technique. One such work is [Ker98], in which a round-based proactive FEC approach is combined with a receiver-based hierarchical recovery scheme.

7 Conclusions

We have demonstrated three fundamental ways in which parity encoding can be used to improve reliable delivery of data to multiple receivers, where the receivers have heterogeneous loss and delay characteristics. Our examination took into account two features of the Internet that can have a significant effect on protocol performance are commonly not considered: the impact of bursty loss on performance, and the effects of lossy feedback. We showed that proactive FEC techniques can alleviate the feedback implosion problem which is a major concern in large-scale reliable multicast protocols. At the same time,

the technique reduces delivery latency without significant increases in the aggregate bandwidth that the sender injects into the network. We demonstrated how receivers could apply proactive FEC techniques to meet real-time deadlines up to a desired probability, and how they could solicit repairs from the sender in an efficient manner when their round trip times differ.

References

[Bla83] R.E. Blahut. *Theory and Practice of Error Control Codes.* Addison-Wesley, Reading, MA, 1983.

[BTW94] J. Bolot, T. Turletti, and I. Wakeman. Scalable Feedback Control for Multicast Video Distribution in the Internet. In *Proceedings of ACM SIGCOMM'94*, pages 58–67, London, U.K., August 1994.

[DL95] H.R. Deng and M.L. Lin. A Type I Hybrid ARQ System with Adaptive Code Rates. *IEEE Transactions on Communications*, COM-43(2/3/4):733–737, Feb/Mar/Apr 1995.

[DO97] D. DeLucia and K. Obraczka. Multicast Feedback Suppression Using Representatives. In *Proceedings of INFOCOM'97*, Kobe, Japan, March 1997.

[FJL+97] S. Floyd, V. Jacobson, C. Liu, S. McCanne, and L. Zhang. A Reliable Multicast Framework for Light-Weight Sessions and Application Level Framing. *IEEE/ACM Transactions on Networking*, 5(6):784–803, December 1997.

[Ker98] R. Kermode. Scoped Hybrid Automatic Repeat Request with Forward Error Correction (SHARQFEC). In *Proceedings of SIGCOMM'98*, Vancouver, Canada, September 1998.

[KKT98] S. Kasera, J. Kurose, and D. Towsley. A Comparison of Server-Based and Receiver-Based Local Recovery Approaches for Scalable Reliable Multicast. In *Proceedings of INFOCOM'98*, San Francisco, CA, March 1998.

[LC99] D. Li and D. Cheriton. Evaluating the Utility of FEC with Reliable Multicast. In *Proceedings of ICNP'99*, Toronto, Canada, October 1999.

[LDW97] M.T. Lucas, B.J. Dempsey, and A.C. Weaver. MESH: Distributed Error Recovery for Multimedia Streams in Wide-Area Multicast. In *Proceedings of IC3N'97*, 1997.

[LLGLA96] B. Levine, D. Lavo, and J.J. Garcia-Luna-Aceves. The Case for Concurrent Reliable Multicasting Using Shared Ack Trees. In *Proceedings of ACM Multimedia'96*, Boston, MA, November 1996.

[LPA99] X. Li, S. Paul, and M. Ammar. Layered Video Multicast with Retransmissions (LVMR): Evaluation of Hierarchical Rate Control. In *Proceedings of IEEE INFOCOM'99*, New York, NY, March 1999.

[LPGLA98] B. N. Levine, S. Paul, and J.J. Garcia-Luna-Aceves. Organizing multicast receivers deterministically according to packet-loss correlation. In *Proceedings of ACM Multimedia'98*, Bristol, U.K., September 1998.

[McA90] A.J. McAuley. Reliable Broadband Communication Using a Burst Erasure Correcting Code. In *Proceedings of SIGCOMM'90*, pages 297–306, Philadelphia, PA, September 1990.

[MJV96] S. McCanne, V. Jacobson, and M. Vetterli. Receiver Driven Layered Multicast. In *Proceedings of SIGCOMM'96*, Stanford, CA, August 1996.

[NB99] J. Nonnenmacher and E. Biersack. Scalable Feedbackfor Large Groups. *Transactions on Networking*, June 1999.

[NBT98] J. Nonnenmacher, E. Biersack, and D. Towsley. Parity-Based Loss Recovery for Reliable Multicast Tranmission. *Transactions on Networking*, August 1998.

[NLJ+98] J. Nonnenmacher, M. Lacher, M. Jung, E. Biersack, and G. Carle. How Bad is Reliable Multicast without Local Recovery. In *Proceedings of INFOCOM'98*, San Francisco, CA, March 1998.

[PPV98] C. Pappadopoulos, G. Parulkar, and G. Varghese. An Error Control Scheme for Large-Scale Multicast Applications. In *Proceedings of INFOCOM'98*, San Francisco, CA, March 1998.

[PSLS97] S. Paul, K.K. Sabnani, J.C. Lin, and S.Bhattacharyya. Reliable Multicast Transport Protocol (RMTP). *IEEE JSAC*, 15(3):407–421, April 1997.

[Riz97] L. Rizzo. Effective Erasure Codes for Reliable Computer Communication Protocols. *Computer Communication Review*, April 1997.

[RJL+00] I. Rhee, S. Joshi, M. Lee, S. Muthukrishnan, and V. Ozdemir. Layered Multicast Recovery. In *Proceedings of IEEE INFOCOM'00*, Tel-Aviv, Israel, March 2000.

[RKT98] D. Rubenstein, J. Kurose, and D. Towsley. Real-Time Reliable Multicast Using Proactive Forward Error Correction. In *Proceedings of NOSSDAV'98*, Cambridge, U.K., July 1998. An extended version appears as UMass CMPSCI technical report 98-19.

[RKTK99] D. Rubenstein, S. Kasera, D. Towsley, and J. Kurose. Improving Reliable Multicast Using Active Parity Encoding Services (APES). In *Proceedings of IEEE INFOCOM'99*, New York, NY, March 1999. An extended version appears as UMass CMPSCI technical report 98-79.

[RV98] L. Rizzo and L. Vicisano. RMDP: An FEC-based Reliable Multicast Protocol for Wireless Environments. *Mobile Computing and Communications Review*, 2(2), April 1998.

[SBE+98] T. Speakman, N. Bhaskar, R. Edmonstone, D. Farinacci, S. Lin, A. Tweedly, L. Vicisano, and J. Gemmell. PGM Reliable Transport Protocol. Internet Draft draft-speakman-pgm-spec-03.txt, January 1998. Expired on December 24, 1999.

[TKP97] D. Towsley, J. Kurose, and S. Pingali. A Comparison of Sender-Initiated and Receiver-Initiated Reliable Multicast Protocols. *IEEE JSAC*, 15(3):398–406, April 1997.

[XMZ97] R.X. Xu, A.C. Meyers, and H. Zhang. Resilient Multicast Support for Continuous Media Applications. In *Proceedings of NOSSDAV'97*, St. Louis, MO, July 1997.

[YKT96] M. Yajnik, J. Kurose, and D. Towsley. Packet Loss Correlation in the MBone Multicast Network. In *Proceedings of the IEEE Global Internet Conference*, London, UK, November 1996.

[YMKT99] M. Yajnik, S.B. Moon, J. Kurose, and D. Towsley. Measurement and Modeling of the Temporal Dependence in Packet Loss. In *Proceedings of IEEE INFOCOM'99*, New York, NY, March 1999.