Improving Reliable Multicast Using Active Parity Encoding Services (APES)*

Dan Rubenstein, Sneha Kasera, Don Towsley, and Jim Kurose
Computer Science Department
University of Massachusetts at Amherst

Technical Report 98-79
Department of Computer Science
July, 1998

Abstract

Traditional reliable multicast protocols for the Internet require large amounts of bandwidth to support reliable delivery of data to numerous receivers. In this paper, we propose and evaluate novel protocols that combine active repair service (a.k.a. local recovery) and parity encoding (a.k.a. forward error correction or FEC) techniques. We show that compared to other repair service protocols, our protocols require less buffer inside the network, and maintain the low bandwidth requirements of previously proposed repair service / FEC combination protocols. Our protocols also reduce the amount of FEC processing at repair servers, moving more of this processing to the end-hosts. Last, we examine repair service / FEC combination protocols in an environment where loss rates are different across domains within the network. We find that in such environments, repair services are more effective than FEC at reducing bandwidth utilization. Furthermore, adding FEC to a repair services protocol not only reduces buffer requirements at repair servers, but also reduces bandwidth utilization in domains with high loss, or in domains with large populations of receivers.

Keywords: reliable multicast, forward error correction (FEC), repair services, active services, performance analysis

1 Introduction

Many applications require reliable multicast delivery from a single sender to a large number of receivers. Providing reliable multicast in a best-effort network where packet losses are frequent, such as the Internet, requires a *reliable multicast protocol*, whose specific function is to compensate for this loss. Because network bandwidth is a limited resource, there is considerable interest in improving upon the bandwidth utilization of such protocols, especially in the event that they must support delivery to thousands of receivers.

Two approaches that have recently shown promise are *repair services* and *hybrid parity encoding / automatic repeat request* (FEC/ARQ for short). The repair services approach makes use of repair servers, located within the network, which buffer data, process feedback from receivers, and retransmit buffered repairs. This approach requires an increase in buffering resources within the network. The FEC/ARQ approach transmits specially encoded repair packets in response to lost data, which must be decoded to retrieve the original data.

To further reduce bandwidth requirements, researchers have considered combining the two approaches into a single protocol. We refer to such combination protocols as *Active Parity Encoding Services* protocols,

*This material was supported in part by the National Science Foundation under Grant No. NCR-9508274, and NCR-9527163, and by DARPA under Grant No. N66001-97-C-8513. Any opinions, findings, and conclusions or recommendations expressed in this material are those of the authors and do not necessarily reflect the views of the National Science Foundation.

†A shorter version of this report will appear in IEEE Infocom '99.

or *APES* protocols for short. These protocols have been shown to be more efficient in terms of network bandwidth usage. However, if not designed carefully, the resulting increase in buffering requirements inside the network can bottleneck overall performance. Because buffer is a resource that will be shared among numerous flows inside the network, it is important to reduce its utilization on a per-flow basis.

In this paper, we introduce two novel APES protocols. These protocols reduce buffering and FEC processing within the network in comparison to previous APES protocols. The reduction in buffer within the network is achieved by restricting buffering to a small set of encoded repairs instead of buffering original data packets. This set of FEC repair packets can be used to repair many combinations of losses incurred by receivers, and because the set is smaller than the set of original data packets, less buffer is utilized. FEC processing is also reduced within the network by keeping the majority of such processing at the end-hosts of the network (sender and receivers).

The first protocol, named **Build-Repairs-Store-Repairs** (BRSR for short), allows for some FEC encoding at the repair servers, but does not require any FEC decoding. The other protocol, named **Get-Repairs-Store-Repairs** (GRSR for short), requires no FEC processing at repair servers of any kind. Instead, repairs are built at the sender and forwarded to repair servers. Compared to the BRSR protocol, the GRSR protocol requires less processing at repair servers, but uses more bandwidth on links between the sender and repair servers.

We perform an analytical study of the bandwidth and buffer requirements of these new APES protocols. We find that, compared to previous APES protocols, the BRSR protocol substantially reduces buffer requirements, while bandwidth utilization increases by a negligible amount. We also compare the buffering requirements of our APES protocols to non-FEC-based repair service protocols. We find that the GRSR protocol uses only a fraction of the buffer that is used by a non-FEC repair server, but that the data rate must be high in order for the BRSR protocol to use buffer more efficiently than non-FEC-based repair service protocols. This means that as network capacities increase leading to increased data rates, the BRSR protocol will become more effective at reducing buffer requirements.

Finally, we consider the bandwidth utilization of APES protocols compared to protocols that use only a repair service approach, or only an end-to-end FEC/ARQ approach. We base our network models on studies which indicate that receiver loss rates vary across different regions of the network. We find that in such a model, repair services provide a more significant savings in bandwidth than end-to-end FEC/ARQ. However, we find that by adding FEC to repair services, in addition to reducing buffer requirements, APES protocols use noticeably less bandwidth than non-FEC repair server protocols in high loss regions, or in regions where repair servers service large numbers of receivers.

The remainder of the paper proceeds as follows. Section 2 provides a description of related work. The network topology and APES protocols are presented in Section 3. The protocols' performance is analyzed in Section 4. Section 5 examines which approach (repair services or FEC) is more effective at reducing bandwidth requirements, and where APES protocols are more effective than either approach used alone. We discuss some limitations of our model in Section 6. Finally, we conclude in Section 7.

2 Related Work

In order to provide repairs to large sets of receivers, traditional reliable multicast protocols multicast their retransmissions [12]. As a result, each retransmitted packet is sent to all receivers, including those that have already received the packet.

Repair server (a.k.a. local recovery) approaches place special *repair servers* with buffering, feedback processing, and retransmission capabilities at strategic locations within the network. Each repair server services a local portion of the multicast group, such that feedback and retransmissions can be restricted within that local region. Several ways of implementing repair servers have been considered. Initially, the popular approach was to have designated receivers perform repair services [8]. It was shown in [5] that a more effective approach would be to have special servers co-located near routers perform the repair services, fitting the active services paradigm described in [1]. An alternative repair server approach uses active networking such that routers themselves perform the repair services [7]. Independent of the approach, we refer to the points that incorporate repair services as *active service points*.

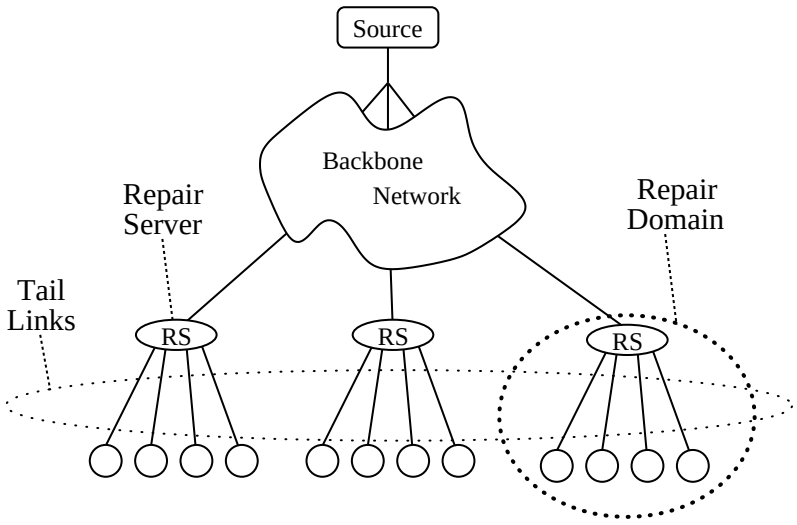

Figure 1: The network model.

The hybrid FEC/ARQ approach involves sending parity packets in place of data retransmissions to recover losses. These packets must be encoded from data packets at the sender, and are decoded upon receipt by receivers to obtain lost data packets. Several efficient encoding techniques are available [11, 2, 3]. It has been shown in [9] that a fixed set of repair packets can recover a larger variety of losses among sets of receivers.

Previous studies of APES approaches have reported improvements in bandwidth performance when compared to end-to-end FEC approaches and non-FEC repair service approaches [10, 6]. Both studies assume encoding, decoding, and limitless buffering capabilities at the active service points. In the next section, we introduce protocols where a main focus is reducing buffer usage. In addition to a lower buffer requirement, these protocols require less encode and decode processing, while maintaining the low bandwidth requirement.

3 APES

Here, we present our model of the network, and three APES protocols that provide efficient, reliable multicast for such a network. The model consists of a multicast tree, which contains a sender at the root of the tree, and receivers at the leaves of the tree.[1] Receivers are topologically partitioned into N *repair domains*, $D_1, \cdots, D_N$. We define the *domain size*, $|D_i|$ to be the number of receivers in domain i. We consider a one level hierarchy of repair servers, where each repair server is located at a point in the tree between the sender and a single domain. This allows the repair server to receive all packets that are multicast from the sender to receivers within its domain. The repair server can also subcast packets to its repair domain, such that these packets are sent down the tree, reaching only the downstream receivers in its repair domain. See Figure 1 for an example.

APES protocols send FEC-based repairs in place of retransmissions. The sender groups data packets into blocks of size k (henceforth k is referred to as the *block size*), and feeds the k data packets into an encoder to produce repair packets. The number of repair packets that can be produced per block is finite. However, this number is sufficiently large that, for our purposes, it is safe to assume that the sender can produce an unlimited supply. Repair packets generated from a block of k data packets are said to belong to the block from which they were generated. Any entity (e.g. repair server or receiver) with a decoder can retrieve the k data packets for a block once it receives any combination of k data and repair packets that belong to that block. FEC-based protocols perform reliable delivery by ensuring that each receiver gets k packets

[1] In practice, what we call a receiver is likely to actually be a local area network (LAN) that can contain several receiving applications.

per block. The fact that receivers can lose different data packets but use the same set of repairs to recover from losses reduces the number of required repair transmissions. This savings was observed for a non-repair server environment, [9].

An APES repair server ensures that each of its downstream receivers gets at least k distinct packets per block. To accomplish this, a repair server performs up to three tasks. First, it can buffer packets (data and/or repairs) sent by the sender. Second, it forwards data packets from the sender to receivers,[2] and can forward sender-generated repair packets if they are needed. Third, armed with an encoder, it can generate additional repairs and buffer and/or send them to receivers. The specific protocol used determines which subset of these three tasks the repair server must perform. Before considering specific protocols, we briefly discuss two ways in which repair servers reduce bandwidth within the network. First, they can prevent unnecessary repairs generated by the source from entering their domain. Second, by locally scoping repairs, additional transmissions to receivers are restricted within its domain, potentially reducing traffic on backbone links as well as within other domains. Bandwidth benefits due to repair servers in a non-FEC context were examined in [5].

We now consider several variations in APES protocols. These variations differ in the manner (how and when) by which additional repair packets are generated by the repair server, and the choice of those packets that are to be buffered by the repair server. In our discussion below, we describe what the repair server does in each protocol in order to ensure that a single block of packets is reliably transmitted to the receivers in its domain. Multiple blocks are delivered by applying the described protocols to each block of data being transferred.

Store-Data-Build-Repairs (SDBR): This protocol is identical to the APES approach presented in [10] and is similar in flavor to the protocol described in [6]. The sender forwards data and FEC packets to repair servers until all repair servers obtain k packets. These k packets are forwarded to receivers, and are also decoded at the repair server to produce the k original data packets, which are buffered by the repair server. When a receiver needs additional repairs, it sends a NAK to the repair server indicating the number of additional repairs it requires. When NAKs arrive at the repair server from receivers requesting additional FEC packets, the repair server itself (rather than the sender) generates new, distinct repair packets, and subcasts them downstream. Because these repair packets are distinct, any receiver that needs additional repairs can use any combination of k data packets and repairs obtained from the repair server.

Build-Repairs-Store-Repairs (BRSR): A repair server decides in advance on a fixed number, b, of repairs per block to generate via FEC encoding. For that block, only these packets are initially buffered and remain buffered for an extended period of time to satisfy receiver losses. The repair server can complete generation of these b repairs for the block once it has successfully received k packets (data and/or repairs) from the sender for the block. These first k packets are referred to as *source packets.* When no packets are lost between the sender and the repair server, the source packets are simply the original k data packets. Otherwise, they include repairs generated by the sender.[3]

To minimize buffering requirements, the repair server does not store the source packets. Each source packet is forwarded downstream to receivers as soon as it is received, and is concurrently input to the FEC encoder. The packet is buffered by the encoder for a short period of while the encoder iteratively applies the packet in the encoding process. The process is iterative because the encoder is able to release each source packet prior to applying the next source packet in the encoding. We refer to this style of encoding as *on-the-fly encoding.* To perform on-the-fly encoding, the repair server must provide space for the b repair packets that will be generated upon arrival of the first source packet. After the kth source packet is fed to the encoder, the b repair packets are ready to be used by receivers to retrieve lost source packets. Some non-trivial issues that result from on-the-fly encoding are discussed briefly in Appendix B.

Prior to being forwarded to receivers, an additional sequence number is prepended to each distinct packet, indicating its order of departure from the repair server. This additional sequence number is referred to as

[2]The assumption that data packets are forwarded by the repair server to receivers simplifies presentation. In practice, to avoid the slow-path, the preferred approach is to have the sender send the data directly to receivers. This can be accomplished by having the repair server send a small control message periodically to receivers. Details appear in Appendix D.

[3]Significant amounts of reordering might also cause the inclusion of repair packets as source packets. In particular, a repair that arrives before a data packet would likely take the place of the data packet as a source packet. Upon arrival at the repair server, the data packet can be treated as lost, or used as a repair.

	Repair Server encoding?	Repair Server decoding?	Repair Server Buffers	Source generated pkts repair rcvr loss?
SDBR	Yes	Yes	Data	Never
BRSR	Yes	No	Repairs	Sometimes
GRSR	No	No	Repairs	Always

Table 1: High level comparison of the three proposed APES protocols.

the *repair server sequence number*, and source packets receive sequence numbers 1 through k. The b repairs that are generated receive repair server sequence numbers $k+1$ through $k+b$. Any subsequent, distinct repairs for the block are assigned a repair server sequence number by the repair server that is one larger than any previous sequence number used for the block.

Receivers send NAKs to recover from losses between the repair server and themselves. A receiver's initial NAK for a block indicates the number of source packets lost. Define l to be the largest number of source packets lost by any receiver in the repair server's domain. As long as $l \leq b$, the repairs in the repair server's buffer are sufficient to repair all downstream receiver losses. Otherwise, the repair server must transmit to receivers $l-b$ additional repairs that are not in its buffer. Because data packets are not stored, the repair server is unable to generate these additional repairs, and must send a NAK to the sender in order to obtain them. After obtaining these additional repairs, the repair server contains a sufficient number of repairs to satisfy all of its downstream receivers. It assigns the new repairs repair server sequence numbers $k+b+1$ through $k+l$, and forwards them to the receivers.

We now point out a key aspect of the protocol that is uncommon in FEC-based protocols. *To make efficient use of the repair server's buffer, the repair packets in the buffer can be retransmitted if they are lost by receivers.* This is done instead of the traditional approach that generates new repairs to satisfy additional losses by receivers. We shall see that this aspect allows for a reduction in buffer, and results in a negligible increase in bandwidth requirements compared to the SDBR protocol. Thus, we still achieve most of the bandwidth efficiency benefits of FEC.

Get-Repairs-Store-Repairs (GRSR): Under this protocol, the repair server does not require FEC encoding capabilities. Instead, it requests b repair packets from the sender, which it buffers and assigns repair server sequence numbers of $k+1$ through $k+b$. Once the repair server obtains the b packets, the protocol behaves identically to the BRSR protocol for the remainder of the transmission of the block.

The value for b can be chosen to be between 0 and the size of the block, k, $(0 \leq b \leq k)$. In Section 4, we examine how the choice of b impacts bandwidth and buffer requirements.

Table 1 presents a high level comparison of the three proposed APES protocols. Protocol SDBR is the most efficient in terms of bandwidth, but must buffer every data packet. Thus, its per block buffer requirements are identical to that of a non-FEC repair server protocol (i.e., one packet is buffered for each data packet sent). It also performs decoding at repair servers.[4] Protocols BRSR and GRSR buffer only repairs. Since receivers reliably receive these repairs, there is never a need to buffer more than k repairs per block. This means that these protocols will never buffer more packets per block than the SDBR protocol. They also do not perform decoding at repair servers.

The BRSR and GRSR protocols differ in that the BRSR protocol requires additional FEC processing at repair servers, while the repair servers in the GRSR protocol perform no encoding. As a result, the GRSR protocol requires extra bandwidth to deliver repairs between the sender and repair server that the BRSR protocol generates with its encoder.

We must also specify what the SDBR and BRSR protocols do with the additional repairs required beyond the b initially stored at the repair server. Such repairs will be sent to a repair server when some receiver in its domain loses more than b source packets. We note three possible options:

[4]The decoding is not a requirement of the protocol. However, previous work assumes the capability is available. If decoding is not performed, then protocol SDBR will require receivers to perform the recursive decoding algorithm described in Appendix B.

Add The additional packets are added to the buffer.

Drop The additional repairs are only forwarded and are not buffered.

Replace The additional repairs are buffered, and older repairs for the same block are removed.

Using the Add policy, the repair server's buffer will contain a sufficient number of repairs to satisfy all receivers once it has expanded to include the additional repairs. For the other policies, the repairs that remain in the buffer might again be inadequate to satisfy a receiver that has lost repairs, and the repair server would again have to contact the sender for retransmission of these repairs.

We have not addressed how the repair server reliably obtains the packets it needs from the sender. We assume that a repair server drops any packet that it did not specifically request (i.e., if it was received as a result of the sender responding to feedback from some other repair server). We note that the buffer and processing requirements as well as the bandwidth between the repair server and receivers in its domain are unaffected by how the sender and repair server decide to reliably deliver data. Hence, to simplify presentation, we assume there is no loss from the sender to the repair server.

In this paper, we only consider a single hierarchical level of repair servers. However, the protocols could easily be extended to include multiple hierarchical levels as follows: in an n-level hierarchy, each repair server in the ith level of the hierarchy provides repair services for a set of repair servers in the $i+1$st level of the hierarchy, $1 \leq i < n$.

4 Buffer/Bandwidth Performance of APES Protocols

As a consequence of our model, each repair domain can be analyzed separately. Let us assume that a block size of k is used, and that a repair server has r receivers downstream, each receiver loses any packet sent to it with a probability p. We consider four performance metrics: the expected number of packet transmissions from the repair server to its downstream receivers, the expected number of repairs received by the repair server from the sender, the expected number of packets that must be buffered per block at a repair server, and the repair server's expected buffer utilization at any moment in time, which we refer to as the *buffer size*.

We begin by constructing an analysis for the BRSR and GRSR protocols. Recall that the k source packets sent from the repair server to receivers are assigned repair server sequence numbers between $1, \cdots, k$. Additional packets, which receivers use to repair losses of source packets, are given sequence numbers larger than k. We assume that the repair server multicasts all packet transmissions. Our analysis also assumes that a receiver that loses m of k packets in a block *requires* the repair server to reliably transmit packets $k+1$ through $k+m$. In practice, a receiver that requires m repairs and loses some of the repairs numbered $k+1$ through $k+m$ could effectively use any repair numbered $k+m', m' > m$ in place of a lost repair. By doing this, the receiver can only decrease the number of times it requests a particular repair. Thus, our assumption gives a conservative upper bound on the number of transmissions from a repair server.

Define ϕ_i to be the probability that a receiver requires transmission of packet i within the block. For $i \leq k$, $\phi_i = 1$. For $i > k$, this is the probability that fewer than $2k - i + 1$ of the initial k packets are received by a receiver. If exactly $2k - i + 1$ of the initial k packets are received, then it can recover the block by reliably obtaining the $i - k - 1$ repairs with sequence numbers $k+1$ through $i-1$. Since repairs with sequence numbers larger than $2k$ are never needed, we have that $\phi_i = 0$ for $i > 2k$. Thus,

$$\phi_i = \left\{ \begin{array}{ll} 1, & i \leq k \\ 1 - \sum_{j=0}^{i-k-1} \binom{k}{j} p^j (1-p)^{k-j}, & k < i \leq 2k \\ 0, & i > 2k \end{array} \right\} \quad (1)$$

Define $q_i(j)$ to be the probability that at least one receiver needs more than j transmissions of packet i,

$$q_i(j) = \left\{ \begin{array}{ll} 1, & i \leq k, j = 0 \\ 0, & i \leq k, j > 0 \\ 1 - (1 - \phi_i p^j)^r, & k < i \leq 2k, j \geq 0 \\ 0, & i > 2k, j \geq 0 \end{array} \right\}$$

4.1 Bandwidth from Repair Server to Receivers

We now consider the expected bandwidth required between the repair server and its receivers (data plus repairs). Let T_{SDBR}, T_{BRSR}, and T_{GRSR} be random variables that denote the number of packets that are transmitted by the repair server using protocols SDBR, BRSR, and GRSR, respectively. The analysis presented in [9] that gives the bandwidth requirements between a sender and a set of receivers over a star topology lends itself directly to the bandwidth computation for the SDBR protocol in our network model:

$$E[T_{SDBR}] = \sum_{j=0}^{\infty} 1 - \left[1 - \sum_{m=0}^{k-1} \binom{k+j}{m} (1-p)^m p^{k+j-m}\right]^r. \quad (2)$$

We now compute upper bounds on $E[T_{BRSR}]$ and $E[T_{GRSR}]$. Let τ_i be a random variable that equals the number of times that packet i is transmitted. For $i \leq k$, we have $E[\tau_i] = 1$, since the packet is always transmitted at most once. For $k < i \leq 2k$, we have

$$E[\tau_i] = \sum_{j=0}^{\infty} P(\text{some receiver needs more than } j \text{ transmissions of packet } i) = \sum_{j=0}^{\infty} q_i(j). \quad (3)$$

Then the expected total number of transmissions that the repair server must make using protocols BRSR and GRSR is

$$E[T_{BRSR}] = E[T_{GRSR}] = k + \sum_{i=k+1}^{2k} E[\tau_i]. \quad (4)$$

We remind the reader that because we assumed each receiver requires reliable transmission of a fixed set of repair packets, equations (3) and (4) give upper bounds on the expected number of packet transmissions that would occur in practice.

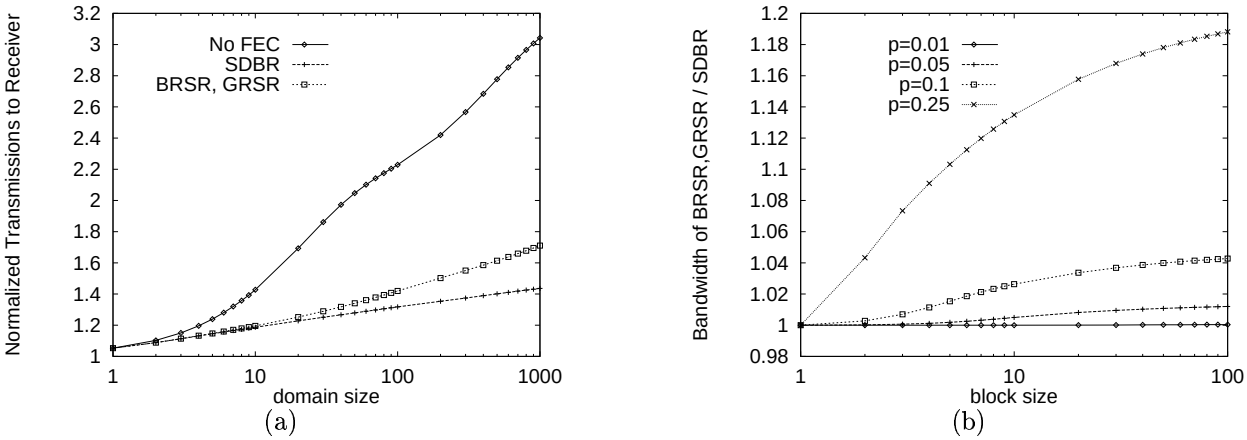

Figure 2: A comparison of protocols with (a) varying domain size and (b) varying block size.

Figure 2(a) presents the expected number of transmissions (normalized per packet) for a block size of 10 and receivers with loss rates of 5% ($p = .05$), as a function of the domain size. We observe a clear reduction in bandwidth due to the introduction of FEC, and further observe that protocols BRSR and GRSR use essentially the same bandwidth as the SDBR protocol until the domain size grows very large. We emphasize that the bandwidth from the repair server to receivers is unaffected by the choice of b, or the buffer policy (i.e., Add, Drop, or Replace).

Figure 2(b) gives the ratio of the upper bound on the expected number of per packet transmissions for BRSR and GRSR protocols over the expected number of per packet transmissions for the SDBR protocol as a function of block size. Here, there are 8 receivers in the repair domain. Curves are presented for various loss rates. We observe that, for loss rates at or below 10%, there is less than a 4% loss of bandwidth.

The difference becomes noticeable as the loss rates increase. Thus, protocols BRSR and GRSR do not use substantially more bandwidth than protocol SDBR between the repair server and receivers for reasonable loss rates.

We point out that the SDBR protocol provides a lower bound on the expected bandwidth for BRSR and GRSR protocols. This is because the SDBR protocol always sends the minimal number of distinct repairs in order for receivers to retrieve the source packets. Thus, the small difference in bandwidth required by protocol SDBR when compared to the upper bounds of protocols BRSR and GRSR indicates that the upper bound is tight. The proximity of the upper and lower bounds also tells us that there would be little bandwidth improvement if the repair server tried to optimize the set of repairs being transmitted. To summarize, *the SDBR protocol requires less bandwidth between the repair server and receivers than protocols BRSR and GRSR. However, for domain sizes and loss rates that one might expect in practice, the difference in bandwidth is negligible.*

4.2 Loss-free bandwidth from sender to repair server

Because protocols BRSR and GRSR buffer a limited number, b, of repairs, the repairs in the buffer at the repair server are insufficient for providing repair for a receiver that loses more than b source packets. Furthermore, by not buffering source packets, the repair server is unable to generate additional repairs, and must obtain additional repairs for receivers from the sender.

Let A_ρ be a random variable equal to the number of additional transmissions the sender must make to a repair server for policy ρ. The approach used to derive equation (4) yields the following result for the Drop policy:

$$E[A_{Drop}] = \sum_{i=k+b+1}^{2k} E[\tau_i]. \tag{5}$$

The expected number of repairs that the sender must reliably send to a particular repair server under protocol BRSR using the Drop policy $E[A_{Drop}]$. For protocol GRSR, where the b buffered repairs are transmitted from the sender as well, the expected number of repairs using policy Drop is $b + E[A_{Drop}]$.

The choice of policy affects the size of the buffer, and the bandwidth requirements between the source and repair servers. We recommend the Drop policy, because it is simple, and compared to the Add policy, it keeps buffer requirements lower. Furthermore, for sufficiently large values of b, there is a negligible increase in bandwidth requirements . The interested reader can see Appendix A for the analysis which leads us to this conclusion.

We now focus on how the choice of b affects per-block buffer and sender-to-repair server bandwidth requirements. The graphs in Figure 3 illustrate the expected, average, additional bandwidth normalized per packet (i.e. $E[A_{Drop}]/k$) as a function of the normalized initial buffer size (i.e., b/k). Unless stated otherwise, a default loss rate of 5%, a block size of 10, and a domain size of 8 are used. These graphs illustrate that the number of times that the repair server must retrieve a repair from the sender is extremely small when b is chosen sufficiently large. From this, we conclude that the additional bandwidth needed from the sender under the Drop policy is negligible.

4.3 Expected Buffer Size

We refer to the size of the buffer being utilized by the protocol at any given time as the *buffer size.* The expected buffer size is an increasing function of both the number of packets that must be buffered, and the amount of time that each packet is buffered. We have shown thus-far that using FEC can reduce the expected number of packets per block that need buffering. We now consider the length of time that packets need to be stored at the repair server under the various APES and non-FEC repair server protocols, and examine the impact that this has on the expected total buffer size. We define B_{RS} to be the buffer size of a non-FEC repair server protocol, and $B_{BRSR-Drop}$ to be the buffer size of protocol BRSR using the Drop policy. We now compute $E[B_{RS}]$ and $E[B_{BRSR-Drop}]$. Later, we discuss how these values compare with the buffer size of the GRSR protocol, $E[B_{GRSR-Drop}]$.

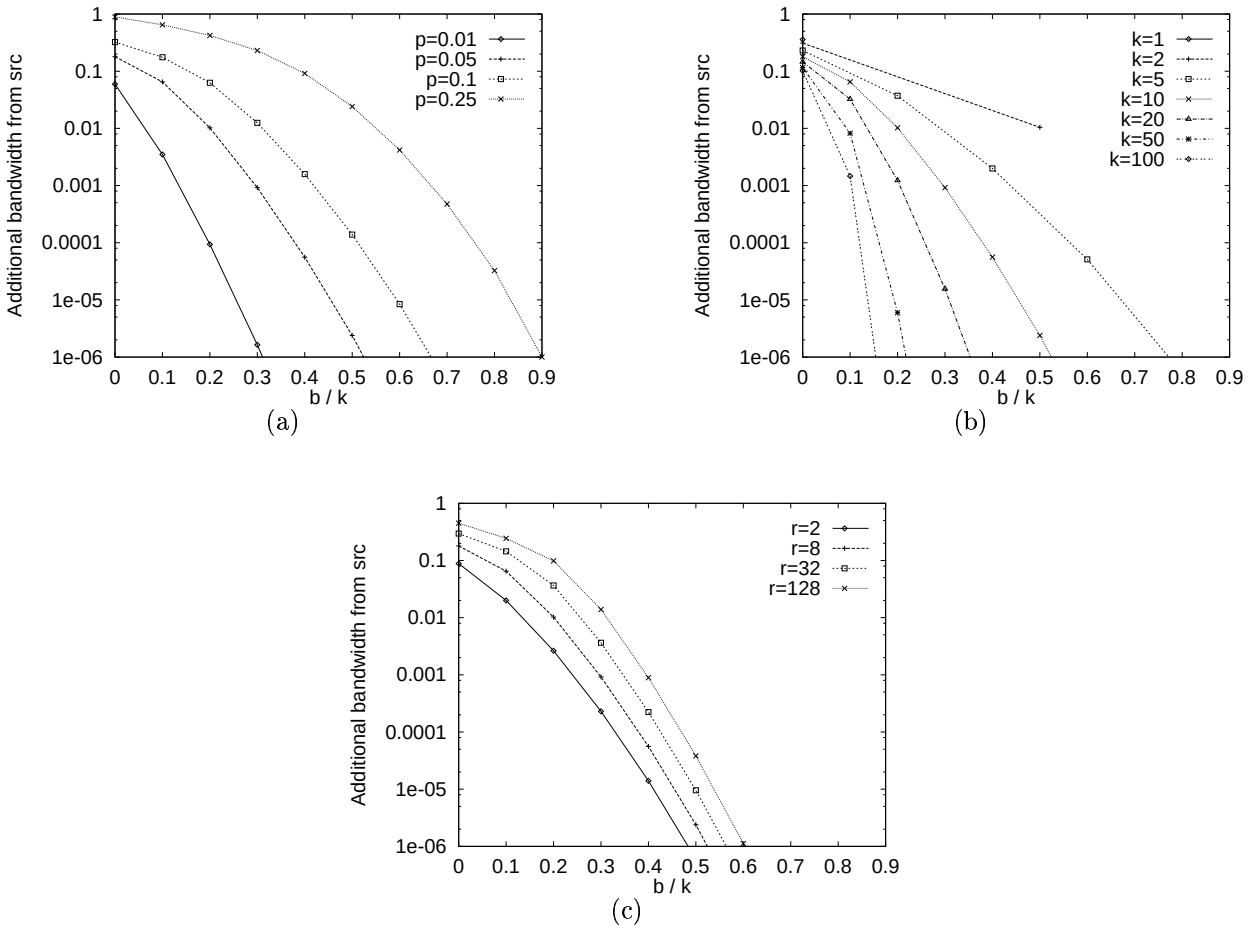

Figure 3: Per packet, normalized, expected number of packets that must be retrieved from sender for protocols Build-Repairs-Store-Repairs and Get-Repairs-Store-Repairs using the Drop policy, due to insufficient buffering at the repair server vs. default size of b for various (a) loss rates, (b) block sizes, and (c) number of receivers in a repair domain.

Our analysis of the BRSR-Drop (RS) protocol considers a repair server (set of receivers) to which data packets arrive with mean arrival rate, Λ. We also assume that there is a fixed round trip time (RTT), R, between each receiver and its upstream repair server, which includes the time that it takes for a receiver to detect a packet loss, send a NAK to the sender, and have the sender retrieve a packet from its buffer and send it to the receiver. When a repair server cannot provide a repair to a receiver, we say that a *repair server miss* has occurred. We measure the repair server's effectiveness in terms of the *repair server miss probability*, $\mathcal{P}_k$, which is defined to be the probability that at least one repair server miss occurs for a block. For the non-FEC protocol, RS, we measure the repair miss probability in terms of $\mathcal{P}_1$, the probability that a repair server miss occurs for a packet.

There are two scenarios under which a miss can occur using the BRSR-Drop protocol: either b repairs are insufficient for some receiver, or one of the b repairs was released from the buffer before it was received by some receiver that required it.

The buffer requirements that would allow a repair server to have a miss probability lower than an application chosen bound, ϵ_1, were computed in [5] for a non-FEC repair server protocol. The repair server maintains each original data packet in its buffer for a fixed number of transmissions, N, which allows each receiver at least N attempts at receiving the packet. The value of N is chosen so that $\mathcal{P}_1 = 1-(1-p^N)^r < \epsilon_1$, where p is the loss probability from the repair server to each receiver, and r is the domain size. Using Little's Law, the expected buffer size can be computed:

$$E[B_{RS}] \quad = \quad \Lambda R(N-1). \tag{6}$$

The value, ΛR indicates the number of packets that a sender is expected to send between receiver retransmission requests, We refer to this value as the *retransmission factor.*

For protocol BRSR-Drop, we compute an upper bound on the time that each packet is maintained in the buffer, such that the repair server miss probability, $\mathcal{P}_k$, is less than an application chosen bound, ϵ_k, for a block size of k. Let N_i be the number of times that a receiver can request transmission of packet i (waiting one RTT between transmission requests) before the repair server drops packet i from its buffer. We have $N_i = 0$ for $i \leq k$. Because we are using the Drop policy, we also have $N_i = 0$, $i > k + b$. We now compute values of N_i, for $k < i \leq k + b$. Define $\psi_R(j)$ to be the probability that receiver R loses j source packets and makes a request for a repair that is not in the buffer. This will occur if $j > b$, or if both $j \leq b$ and some packet $k + m$ is not received after N_{k+m} transmissions, $1 \leq m \leq j$,

$$\psi_R(j) \quad = \quad \left\{ \begin{array}{ll} \binom{k}{j} p^j (1-p)^{k-j} \left[1 - \prod_{m=1}^{j} (1 - p^{N_{k+m}}) \right] & 1 \leq j \leq b \\ \binom{k}{j} p^j (1-p)^{k-j} & j > b \end{array} \right\} \tag{7}$$

Define ψ_R, to be the probability that receiver R requires a packet that causes the repair server to request an additional packet from the sender.

$$\psi_R \quad = \quad \sum_{j=1}^{k} \psi_R(j) \tag{8}$$

The repair server miss probability, $\mathcal{P}_k$, equals the probability that a repair server needs to request an additional repair from the sender. This equals the probability that at least one downstream receiver requires such a packet.

$$\mathcal{P}_k \quad = \quad 1 - (1 - \psi_R)^r \tag{9}$$

Thus, it is sufficient for a repair server to choose b and the set of $\{N_i\}$ sufficiently large so that $\mathcal{P}_k < \epsilon_k$.

We now determine the expected buffer size. Each repair packet i, $k < i \leq k + b$, begins to occupy buffer space once the first source packet for the block arrives at the repair server. To be conservative, we assume that no NAKs are sent from receivers until the kth source packet has been transmitted by the repair server, and has been given ample time (e.g., half a RTT) to be received.[5] Afterwards, the ith repair packet is held for a period of time that allows receivers N_i attempts at retrieving it. The rate at which packets with repair server sequence number i arrive (or are constructed) at the repair server is Λ/k. Buffer space is used by the ith packet during the period of time it is being constructed (i.e., the time between the arrival of the first and kth source packet), plus enough time to allow N_i retransmissions to receivers. The expected amount of time that packet i resides in the buffer is $(k-1)/\Lambda + RN_i$. It follows from Little's Law that the expected amount of buffer being used to hold packet i over all blocks is $(k-1)/k + \Lambda R N_i/k$. The total expected buffer size is obtained by summing over the b values of i for which packets can reside in the buffer:

$$E[B_{BRSR}] \quad = \quad \frac{b(k-1)}{k} + \frac{\Lambda R}{k} \sum_{i=k+1}^{k+b} N_i. \tag{10}$$

We compare buffer size between an APES protocol and a non-FEC protocol by choosing a bound on the per-packet repair server miss probability, ϵ_1. Unfortunately, we cannot compute the per-packet repair server miss probability for an APES protocol, since misses occur in blocks. We note, however, that the non-FEC protocol can measure its miss probability in terms of blocks of k packets, $\mathcal{P}_k$. Since receiver losses are spatially and temporally independent in our network model, achieving a per-packet repair server miss probability below a bound of ϵ_1 is identical to achieving a per-block repair server miss probability below a bound of ϵ_k for a block size of k when

[5] Note that receivers could potentially send NAKs sooner. For example, a loss of the first source packet could be detected long before the kth source packet arrives.

$$\epsilon_k \quad = \quad 1 - (1 - \epsilon_1)^k. \tag{11}$$

We compute the expected buffer size of protocol BRSR using a per-packet repair server miss probability by using (11) to convert the per-packet repair server miss probability to a per-block repair server miss probability, and compute values for b and $\{N_i\}$ such that $\mathcal{P}_k < \epsilon_k$, and use these values in equation (10).

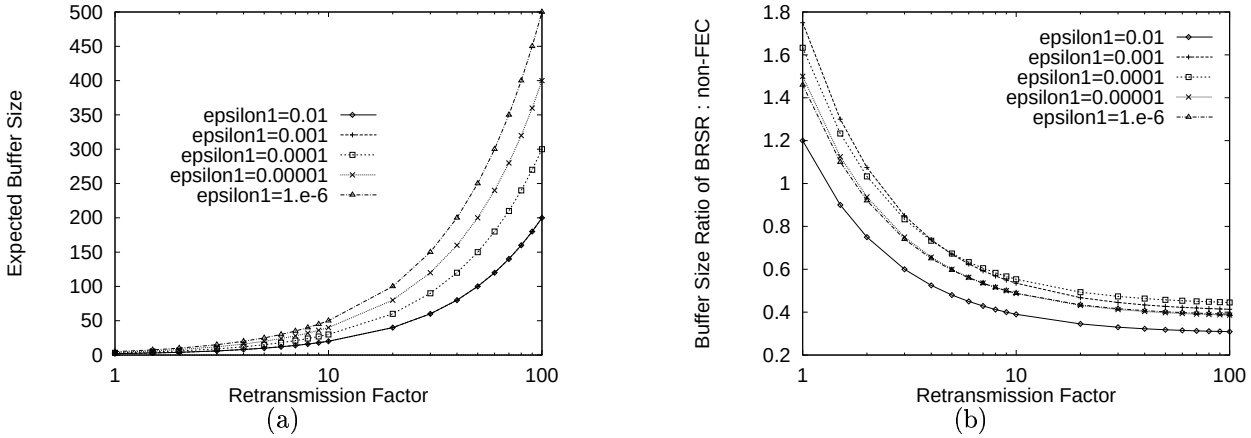

Figure 4: (a) Expected Buffer Size for flows using non-FEC repair services, (b) Ratio of expected buffers of BRSR to non-FEC.

Figure 4 demonstrates how the retransmission factor affects the expected buffer size for various values of ϵ_1. The x-axis indicates the retransmission number, and the various curves represent the values of ϵ_1. In this figure, the loss rate is .05, and the domain size is 8. Figure 4(a) gives the expected amount of buffer for a non-FEC repair server protocol. We see that for a small retransmission factor, very little buffer is required. The buffer size increases linearly with the retransmission factor.

Figure 4(b) gives the ratio of the expected buffer of the BRSR protocol with a block size of 10 to that of the non-FEC protocol. We see that for a small retransmission factor, the expected buffer of the non-FEC protocol is actually less than that of the BRSR protocol. This is due to the fact that in the non-FEC protocol, a data packet is buffered for less time than the period of time between the sender's transmission of the first and last packets that would make up a block if FEC were used. For the BRSR protocol, no repairs can be sent until all packets for a block are received.[6] Thus, very low rate data transfers make inefficient use of the buffer. However, we see that as the retransmission factor increases, the BRSR protocol's buffer increases at a slower rate, and buffer size is considerably smaller for high rate data flows.

Varying the loss rate or the domain size does not have a predictable effect on the ratio of the buffer size of protocol BRSR to buffer size of a non-FEC protocol. Increasing either parameter increases the repair miss probability. However, the buffer size only changes when the repair server miss probability increases beyond the fixed bound, ϵ_i. At this point, the buffer size increases. Thus, the buffer size as a function of loss rate or domain size is a non-continuous, non-decreasing step function. Since these steps occur at different points for the BRSR and RS protocols, the ratio of buffer sizes as a function of loss rate or domain size is a non-continuous step function where steps can either increase or decrease.

Figure 5 gives the expected buffer size for protocol BRSR as a function of the retransmission factor for a variety of block sizes. We see that choosing a block size to minimize the expected buffer size depends on the retransmission factor. For small values of the retransmission factor, a large block size forces a repair server to delay a retransmission while it waits for the arrival of all the source packets for the block. In that same period of time, a smaller block size would have allowed the repair server to perform several retransmission iterations with the receivers, which would allow it to safely release repairs from its buffer sooner. As the

[6]This fact holds true for the SDBR protocol as well, and its expected buffer size is larger for small retransmission factors, as well.

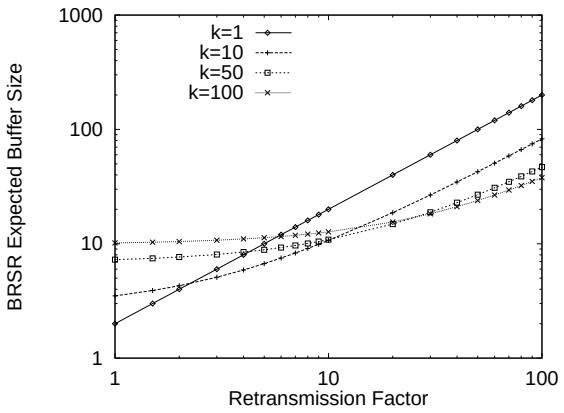

Figure 5: Expected BRSR buffer for various block sizes.

retransmission factor increases, the receiver makes fewer retransmission requests before the repair server obtains the final source packet. As a result, larger block sizes reduce the buffer size due to their repair efficiency.

We have seen that the retransmission factor is a critical factor in determining how effective an FEC approach is at reducing buffer requirements. At today's network speeds, the approach might not provide much savings in buffer. For example, if the RTT between receiver and repair server is 40 ms, and a sender transmits at 15 kilobytes per second, using half-kilobyte packets, the retransmission factor equals 1.2. However, the expectation is that network capacities will increase, while link speeds will remain fixed. Thus, we can expect large increases in sender transmission rates without significant decreases in round trip times. This can dramatically increase the retransmission factor. For instance, if a session runs at a megabit per second uses half-kilobyte packets, and sees an RTT of 40 ms between receiver and repair server (corresponding to a retransmission factor of 5.12), then we expect to see between 20% and 50% reduction in buffer for reasonable domain sizes and loss rates by using an APES protocol in place of a non-FEC protocol.

We point out several additional factors that favor an APES approach.

- Our computations for the buffer used by an APES approach are conservative in that we assume receivers wait to send their NAKs only after all source packets for a block have been transmitted. If receivers instead send NAKs as soon as they detect loss, we expect a reduction in expected buffer size, especially for small retransmission factors.

- It is unlikely that the response time of a receiver to a lost packet from a repair server will be as low as the round trip time between the receiver and repair server, since numerous other events, such as additional processing and randomized back-off timers would increase response times. This would increase the value of the RTT estimate, R, increasing the retransmission factor.

- The application might choose to send packets in bursts. By fitting the block size to the size of the burst, the delay between the arrival of the first and last packet in a block is reduced significantly. This would make buffering more efficient using an APES protocol, even for low-rate data transfers.

- Several reliable multicast protocols send loss feedback in batches. One such example is the RMTP protocol [8, 13], which sends a bit-vector indicating which packets have been lost. This approach introduces a delay in feedback for a non-FEC approach. By matching the block size to the rate at which the bit-vector moves forward in the sequence number, there will be no additional buffering due to using FEC packets.

Finally, we discuss the buffer requirements of the GRSR protocol using the Drop policy. We note that the buffer requirements differ in that repair packets are not be cached in advance of receiving the kth source packet. We conservatively assume that all repairs arrive immediately after the last source packet for the block.[7] Little's law gives the expected buffer size to be:

[7]If packets arrive out of order at the repair server (i.e., repairs arrive before source packets), the repair server can send the repairs as source packets and use the late source packets as repairs.

$$E[B_{GRSR}] = \frac{\Lambda R}{k} \sum_{i=k+1}^{k+b} N_i. \quad (12)$$

The ratio of expected buffer size of GRSR to non-FEC repair service remains constant as the retransmission factor varies. Since the values of N and $\{N_i\}$ are chosen independent of the retransmission factor, it turns out that

$$E[B_{GRSR}]/E[B_{RS}] = \lim_{R\Lambda \to \infty} E[B_{BRSR}]/E[B_{RS}]. \quad (13)$$

The ratio of expected buffer size of GRSR to non-FEC repair service can be viewed as the asymptotic values of the curves in Figure 4(b) as the retransmission factor increases.

4.4 Analysis Summary

We have analyzed the buffer and bandwidth requirements for various APES protocols. For domain sizes and loss rates that one would expect in practice, the BRSR protocol bandwidth requirements are similar to the SDBR protocol, and use considerably less buffer. The bandwidth requirements of the GRSR protocol on links between the repair server and receivers are identical to those of protocol BRSR. However, because encoding is not performed at the repair server, additional bandwidth is used between the sender and repair servers to transmit all repairs which are to be buffered. The GRSR protocol uses a fraction of the buffer that a non-FEC repair services protocol uses to meet a fixed miss probability. The BRSR protocol uses more buffer to meet a fixed miss probability for slow data rates. However, as the data rate increases, the buffer requirements of the BRSR protocol approach those of the GRSR protocol, and are considerably less than the buffer requirements of non-FEC repair server protocols.

5 What network environments require APES protocols?

We now consider the reduction in network bandwidth that results from using APES protocols. We also reveal a type of variation that has been observed in real networks that we believe causes the largest variation in the amount of savings in bandwidth that is obtained by using an APES approach.

We extend our network model to consider receivers with different loss rates. Define $\mathcal{R}_{i,j}$ to be the jth receiver in domain i, $1 \leq j \leq |D_i|, 1 \leq i \leq N$. We define $P_{i,j}$ to be the loss probability observed by receiver $\mathcal{R}_{i,j}$. Without loss of generality, we assume that receivers are enumerated within each domain, i, so that for any $1 \leq j < j' \leq |D_i|, P_{i,j} \leq P_{i,j'}$, i.e., receivers are ordered within each domain by increasing loss rate.

We define a multicast session to be *intra-homogeneous* if, for any i and for all $j, j', P_{i,j} = P_{i,j'}$, i.e., all receivers in the same domain have the same loss rate. A session that is not intra-homogeneous is *intra-heterogeneous.* We define a session to be *inter-homogeneous* if for any two domains, D_i and $D_{i'}$, both of the following hold:

- $|D_i| = |D_{i'}|$.
- For each $j \leq |D_i|, P_{i,j} = P_{i',j}$.

If a session is not inter-homogeneous, it is defined to be *inter-heterogeneous.*

We also assume that the sender always multicasts repairs, and the repair server always subcasts repairs. For simplicity, we again consider a network where all losses occur on the tail links, which lie between each receiver and a repair server (again, the reader is referred to Figure 1. This means that receiver losses are independent from one another, and that there are no losses within the backbone network (i.e., no losses between the sender and the repair server).

5.1 Bandwidth Comparison

Nonnenmacher et al showed in [10] that for small block sizes, an APES approach reduces network bandwidth over that of a FEC/ARQ approach in the case of inter-homogeneous, intra-homogeneous sessions, but the reduction becomes less as the block size increases until it is almost unnoticeable for block sizes around 100. They also came to similar conclusions for inter-homogeneous, intra-heterogeneous sessions. However, they did not discuss inter-heterogeneous sessions. Loss studies over the MBone [14, 4] suggest that loss rates differ in different regions of the Internet. Additionally, the domain size can vary considerably over the set of domains in a session. Consequently, we expect most multicast sessions to be inter-heterogeneous.

We now examine an inter-heterogeneous, intra-homogeneous session, in which there are two types of domains. There are n_i domains of type i, where each domain of type i consists of r_i receivers, each having a loss rate of p_i, $i = 1, 2$. In this domain, we consider two protocols.

e-e: The e-e protocol is a hybrid, end-to-end FEC/ARQ protocol. Reliable transmission of data is performed on a block-by-block basis. For each block, the sender transmits the data packet within the block, and awaits NAKs from the receivers. A receiver's NAK indicates the number of additional FEC repairs from the block that it needs in order to perform decoding. If a receiver does not need additional repairs, it need not send a NAK. The sender builds and transmits a number of repairs equal to the maximal request over all receivers. The feedback process repeats itself until all receivers obtain a sufficient number of repairs, and cease sending NAKs.

SDBR: The SDBR protocol is described in Section 3. We use this protocol within the comparison because it is most similar in flavor to e-e, with the exception that repair servers generate repairs instead of the sender. By setting the block size to one, this protocol performs identically to a non-FEC repair server protocol.

Our measure of bandwidth is the number of packets (data plus repairs) that are sent per block of data. We define the random variable, N_i^{SDBR}, to equal the number of packets multicast from a repair server to downstream receivers for a block in the SDBR protocol. We extend equation (2) so that it applies to the various types of domains:

$$E[N_i^{SDBR}] = \sum_{j=0}^{\infty} 1 - \left[1 - \sum_{m=0}^{k-1} \binom{k+j}{m} (1-p_i)^m {p_i}^{k+j-m}\right]^{r_i}, i = 1, 2. \quad (14)$$

We define N^{e-e} to be a random variable that equals the number of (data and repair) transmissions in the e-e protocol that a sender must transmit to all downstream receivers (over all domains). Then

$$E[N^{e-e}] = \sum_{j=0}^{\infty} 1 - \prod_{i=1}^{2} \left[1 - \sum_{m=0}^{k-1} \binom{k+j}{m} (1-p_i)^m {p_i}^{k+j-m}\right]^{r_i n_i}. \quad (15)$$

We also define F_i to a random variable that equals the number of transmissions in a domain of type i by a repair server using a protocol that does not implement FEC. $E[F_i]$ equals $E[N_i^{SDBR}]$ when $k = 1$, and can be written more succinctly as

$$E[F_i] = \sum_{j=0}^{\infty} 1 - (1 - {p_i}^j)^{r_i} \quad (16)$$

Figure 6 compares the number of expected transmissions from a repair server to receivers in its repair domain using an APES approach to non-APES approaches as (a) the block size, k, varies, (b) the domain size varies, and (c) the high loss rate varies along the x-axis. As a default, the block size, k, equals ten, the domain size of each domain, r_i, is ten, and there are ten domains of type one with loss rates from repair server to receiver of $p_1 = .01$, and a single domain of type two with loss rates of $p_2 = .1$. In Figures 6(a) and 6(b), the block size, k, and the domain size vary along the x-axis, respectively. Figure 6(a) plots curves, where the y-axis equals the normalized, expected cost of reliably transmitting a data packet. The curves labeled **e-e**, **apes low**, and **apes high** plot values of $E[N^{e-e}]/k$, $E[N_1^{SDBR}]/k$, and $E[N_2^{SDBR}]/k$, respectively. Note that values for $E[F_1]$ and $E[F_2]$ are given by respective plots **apes low** and **apes high** when the block size,

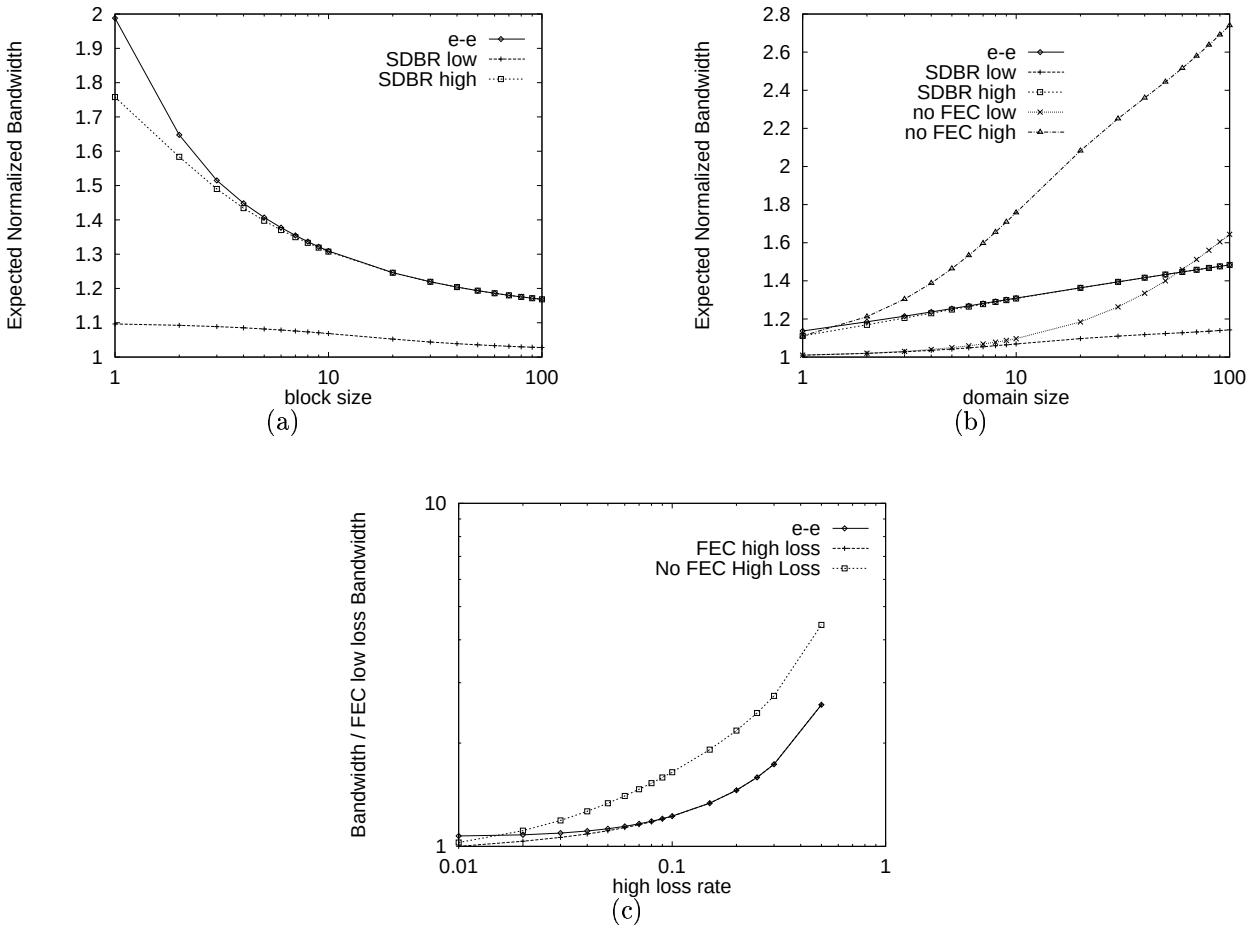

Figure 6: Expected number of packets transmitted to receivers in inter-heterogeneous, intra-homogeneous networks varying (a) block size, (b) repair domain size, and (c) high loss rate.

k, equals one. Figure 6(b) includes additional plots for $E[F_1]$ and $E[F_2]$, respectively labeled **no FEC low** and **no FEC high**. In Figure 6(c), the loss rate in the type two, high loss, domain is varied along the x-axis, while the y-axis equals the normalized, expected cost of reliably transmitting a data packet in multiples of this cost in a low loss (type one) domain. Plots labeled **e-e**, **FEC high loss**, and **No FEC high loss** respectively plot values for $E[N^{e-e}]/E[N_1^{SDBR}]$, $E[N_2^{SDBR}]/E[N_1^{SDBR}]$, and $kE[F_2]/E[N_1^{SDBR}]$.

From Figures 6(a) and 6(b), we see that the bandwidth used by the e-e protocol throughout the network is almost identical to what is required in the high loss domain. Thus, by using an end-to-end FEC protocol, low loss domains incur much more bandwidth usage than what they incur when repair servers are used. Thus, we see that when there are few high loss domains and many low loss domains, using an end-to-end protocol results in a significant amount of wasted bandwidth throughout the network. In Figure 6(c), we see that in an intra-homogeneous, intra-heterogeneous network, there is a small savings in bandwidth due to the use of repair services. As we increase the heterogeneity, we see how a single high loss domain dominates the bandwidth consumption in the e-e protocol. We also see that adding FEC reduces bandwidth further in protocols that make use of repair services, particularly in high loss domains. This observation can be drawn in Figure 6(a) from the decrease in bandwidth as the block size increases, as well as in Figure 6(b) from by comparing the SDBR protocol and the non-FEC protocol for either the high or low loss domain. We also observed little variation in bandwidth usage as the ratio of the number of low loss domains to the number of high loss domains, n_1/n_2, increases. The plot has been omitted due to lack of variation along x-axis.

We can conclude from these observations that *the bandwidth that is used throughout the network in an*

end-to-end FEC approach is dominated by the bandwidth required by domains with high loss, whereas in a network with repair servers, this bandwidth consumption can be limited to the domains where it is required. An additional savings in bandwidth comes as a result of using FEC, particularly in high loss domains or in large domains. Regardless of block size, an APES approach can reduce bandwidth usage in low loss domains. We expect the improvement due to FEC to be less if the loss between repair server and receivers has a high level of spatial correlation. However, our analysis in Section 4 showed that using FEC still improves performance by reducing buffering requirements.

6 Discussion: Network Model Assumptions

We have performed our examination of APES protocols using a simplified network model. One direction for future work is to implement the protocols and compare their performance in a real networking environment. Since we now understand how certain basic characteristics affect performance, another direction is to use a model which more accurately captures a real networking environment. In particular, we have not directly considered loss between the source and repair servers, temporally correlated (bursty) loss, or spatially correlated loss. We point out that loss between the source and repair servers does not affect the bandwidth requirements between repair servers and receivers. Nor does it affect the buffer requirements of repair servers. Results in [9, 4] indicate that the levels of burstiness (i.e., temporally correlated loss) observed in today's Internet will cause a slight, often unnoticeable decrease in performance of FEC.

We now turn our attention to spatially correlated loss. To begin with, determining the amount of spatially correlated loss within the network is still an open issue. A comparison of the effects of spatially correlated loss near the source between non-FEC repair service protocols, end-to-end FEC protocols, and the APES protocol, SDBR, was examined in [10]. Placing repair servers in the network simplifies analysis of this type of spatially correlated loss because such loss only affects bandwidth on the links that lie above the repair servers. We note that repair packets transmitted in an identical manner between the sender and repair servers in protocols SDBR and BRSR. Because spatial loss correlation between the sender and repair servers affects both protocols identically, it does not change the results of our performance comparisons between these two protocols. Also, the results of the spatially correlated analysis in [10] can be applied directly to the BRSR protocol to compare its bandwidth utilization between the sender and repair servers to that of end-to-end FEC protocols and non-FEC repair server protocols. The analysis applies directly to the GRSR protocol as well, by using a block size of $k + b$ in place of k (since this is the number of packets that the sender must deliver reliably to repair servers). We also point out that a domain size of one is equivalent to assuming that all losses within a domain are 100% correlated. Thus, the bandwidth utilized in a within a domain size of r with partially correlated loss lies somewhere between the results plotted for a domain size of r and those plotted for a domain size of one. While correlated loss will reduce the effectiveness of FEC in reducing bandwidth requirements, we can expect an increase in its effectiveness at reducing buffering requirements.

7 Conclusion

There is still considerable interest in reducing the bandwidth consumption of reliable multicast protocols. Two approaches that successfully achieve such reduction are repair services approaches and hybrid FEC/ARQ approaches. We have examined and compared performance of protocols that combine these approaches, which we call APES protocols. We described several new APES protocols that maintain this high bandwidth efficiency while reducing buffer and FEC processing requirements. We thoroughly explored how the bandwidth, buffer, and FEC processing requirements vary among the various versions of the protocols. We also describe the kinds of networking environments in which such protocols improve the bandwidth efficiency of reliable multicast beyond what either approach is able to accomplish separately.

Acknowledgments

We would like to thank Steve Zabele of TASC, who suggested that we consider on-the-fly encoding techniques.

References

[1] E. Amir, S. McCanne, and R.H. Katz, *The Media Gateway Architecture: A Prototype for Active Services*, to appear in ACM SIGCOMM '98.

[2] J. Blomer, M. Kalfane, R. Karp, M. Karpinski, M. Luby, and D. Zuckerman, *An XOR-Based Erasure-Resilient Coding Scheme*, International Computer Sciences Institute Technical Report ICSI TR-95-048, August 1995.

[3] J. Byers, M. Luby, M. Mitzenmacher, and A. Rege, *A Digital Fountain Approach to Reliable Distribution of Bulk Data*, to appear in ACM SIGCOMM'98, Vancouver, CA, September 1998.

[4] M. Handley, *An Examination of MBone Performance*, Technical Report, UCL and ISI, January 1998.

[5] S. Kasera, J. Kurose, D. Towsley, *A Comparison of Server-Based and Receiver-Based Local Recovery Approaches for Scalable Reliable Multicast*, UMass CMPSCI Technical Report 97-69. A shorter version of this report appeared in IEEE INFOCOM98.

[6] R. Kermode, *Scoped Hybrid Automatic Repeat Request with Forward Error Correction (SHARQFEC)*, to appear in ACM SIGCOMM'98, Vancouver, CA, September 1998.

[7] L.H. Lehman, S.J. Garland, and D.L. Tennenhouse, *Active Reliable Multicast*, IEEE INFOCOM98.

[8] J. C. Lin and Sanjoy Paul. *Reliable Multicast Transport Protocol (RMTP)*, IEEE JSAC, 407, 3, 1414-1424,April 1997.

[9] J. Nonnenmacher, E. Biersack, and D. Towsley, *Parity-Based Loss Recovery for Reliable Multicast Transmission*, ACM SIGCOMM '97, Sept. 1997 Cannes, France, pp. 289-300.

[10] J. Nonnenmacher, M. Lacher, M. Jung, E. W. Biersack and Georg Carle, *How bad is reliable multicast without local recovery?*, Proceedings of IEEE INFOCOM'98, San Francisco, CA, USA, March 1998.

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

[12] D. Towsley, J. Kurose, S. Pingali, *A Comparison of Sender-Initiated and Receiver-Initiated Reliable Multicast Protocols*, IEEE Journal on Selected Areas in Communications, April 1997.

[13] B. Whetten et al, *The RMTP-II Protocol*, Internet Draft, April, 1998.

[14] M. Yajnik, J. Kurose, D. Towsley, *Packet Loss Correlation in the MBone Multicast Network*, IEEE Global Internet Conference. London, November 1996.

Appendix

A Why not use the Add policy?

We claim in Section 4 that the Drop policy is better than the Add policy because it uses a negligible amount of additional bandwidth, and uses significantly less buffer. In this section of the Appendix, we quantitatively demonstrate this fact.

For the Add policy, which buffers additional transmissions, the expected number of additional transmissions from the sender is:

$$E[A_{Add}] = \sum_{i=k+b+1}^{2k} q_i(0). \tag{17}$$

The expected number of repairs that the sender must reliably send to a particular repair server under protocol BRSR using the Add policy $E[A_{Add}]$. For protocol GRSR, where the b buffered repairs are transmitted from the sender as well, the expected number of repairs using policy Add is $b + E[A_{Add}]$.

A.1 Number of Packets Buffered per block

We now examine the per block buffering requirements. For protocol SDBR, we know that this buffer requirement equals the block size, k. For protocols BRSR and GRSR, the Drop and Replace policies have buffer requirements equal to b. We now compute the expected buffer requirement of the Add policy.

To simplify notation, we let $q_i = q_i(0)$. If S_{Add} is the buffer size needed to transmit the block, then for the Add policy, we have that

$$\begin{aligned} E[S_{Add}] &= b + \sum_{j=1}^{k-b} j\mathrm{P}(\text{repair server must buffer } j \text{ additional packets}) \\ &= b + \sum_{j=1}^{k-b} j(q_{k+b+j} - q_{k+b+j+1}) = b + \left[\sum_{j=1}^{k-b} q_{k+b+j}\right] - q_{2k+1} = b + \sum_{j=1}^{k-b} q_{k+b+j} \end{aligned} \quad (18)$$

We note that if b is set to 0, then (18) equals the expected number of packets that must be buffered at a repair server to satisfy all downstream receivers.

A.2 Choosing a Policy

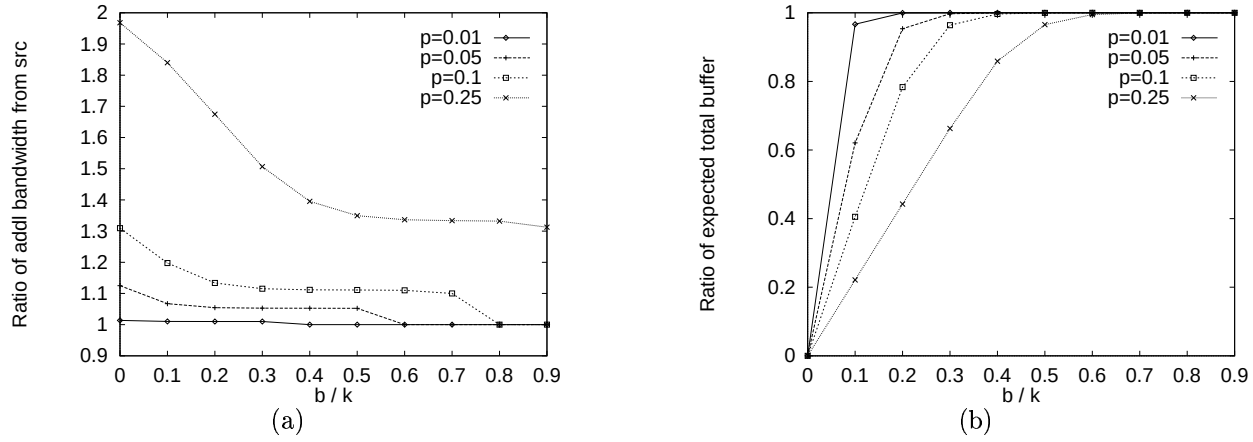

Figure 7: Ratio of (a) additional bandwidth from the sender and (b) expected buffer size for the Drop policy over the Add policy.

Figure 7(a) plots the value of $E[A_{Drop}]/E[A_{Add}]$ for various loss rates as a function of the normalized, initial buffer capacity, b/k. Figure 7(b) plots the ratio of the total expected buffer requirements of the Drop policy compared to that of the Add policy as a function of the initial buffer capacity, b (i.e., $b/E[S_{Add}]$). For both figures, a block size of 10 and a domain size of 8 were used.

The results of Figure 7(a) show us that the bandwidth from the sender using the Drop policy is a small multiple of that used by the Add policy. Because the expected amount of bandwidth is negligible for a reasonably sized default buffer, b, we conclude that the bandwidth savings achieved using the Add policy over the Drop policy is negligible. Figure 7(b) tells us that for a reasonably sized default buffer, the buffer expansion is again a small fraction of the original buffer size. This fractional change in expected buffer size can play a significant reduction in per-block buffer capacity of a repair server. For these reasons, we recommend the Drop policy over the Add policy. The Replace policy has the identical buffer requirements of the Drop policy, and the expected bandwidth cannot be less than that of the Add policy. Thus, we cannot expect much of a gain in performance by using the Replace policy over the Drop policy. *We conclude that either the Drop or Replace policies are best, since they use roughly the same expected bandwidth as the Replace policy, and noticeably less expected, per-block buffer.*

B Implementation Issues

To simplify our presentation, we have avoided discussing several non-trivial issues that arise when considering an implementation of the BRSR and GRSR protocols. We now list a set of issues and briefly describe their ramifications, and possible solutions.

B.1 Encoding / Decoding Issues

We examined two FEC packages that are publically available and found them to be lacking in some functionality required by APES protocols. The first package is described in [11] and uses Vandermonde-based Reed-Solomon codes. The second, described in [2], uses Cauchy-based Reed-Solomon codes. Given a fixed set of data packets, the Cauchy-based encoder cannot be called repeatedly to generate a non-intersecting subsets of repairs. Thus, the sender would have to know up front how many repairs to generate. Both encoders require all data packets simultaneously in order to generate repairs. As a result, they do not directly support on-the-fly encoding. We have modified both packages so that they can perform on-the-fly encoding.

These modifications consist of exchanging the order among certain iterative loops, changing the functional interface to the encoding / decoding functions, and moving or segmenting several initialization functions. One additional byte per block that is in the process of being generated must be maintained by the encoder. None of the changes increase the computational complexity of the encoding / decoding procedures. For more information on the availability of the modified code,

- see **http://www-net.cs.umass.edu/~drubenst/software/software.html**, or
- contact Dan at **drubenst@cs.umass.edu**.

We have performed experiments using the Vandermonde-based encoder which show that our modifications produce a negligible change in the running times of the encoding and decoding processes.

The decoder must be modified to perform *recursive decodes*, i.e. to perform its normal decode operation on a set of received packets, retrieve the source packets, and feed the source packets back into the decoder to retrieve the original data packets. This can be accomplished by repeatedly running an unmodified decoder, or by modifying the decoder so that it performs decoding for several iterations.

B.2 Buffering the generator matrix

In order to support FEC-encoding, the repair server must maintain some additional state, commonly referred to as a generator matrix. The matrix is applied to source packets during encoding operations. A single matrix can support numerous flows, and can support numerous block sizes.[8] Thus, this state is constant with the number of flows, even when the various flows are using different block sizes. Since we only expect buffer to be a limited resource when a repair server supports numerous flows, the per-flow cost of this additional state is minimized.

B.3 Encoding / decoding costs

We omit focusing an analysis on FEC encoding and decoding costs because we do not expect them to be a sufficient bottleneck, as long as block sizes are kept relatively small. However, a simple examination appears in Appendix C, which considers the impact of decode processing in an environment where there is a hierarchy of repair servers. We point out that software implementations of FEC encoding / decoding are sufficiently fast for today's data rates, and research continues to produce faster methods of encoding. Furthermore, if forward error correcting techniques continue to show promise for a large set of applications, it makes sense to design the processing capabilities into hardware. Doing so would dramatically reduce processing costs, and make the approach acceptable for much higher data rates.

[8] The newest version of the Vandermonde-based encoder from [11] constructs a separate generator matrix for each block size. Our modified version of the code allows the use of a single generator matrix for all block sizes. This change affects the space complexity, but not the time complexity of the encoding and decoding operations. Tests have shown that there is little impact on the encoding and decoding speeds.

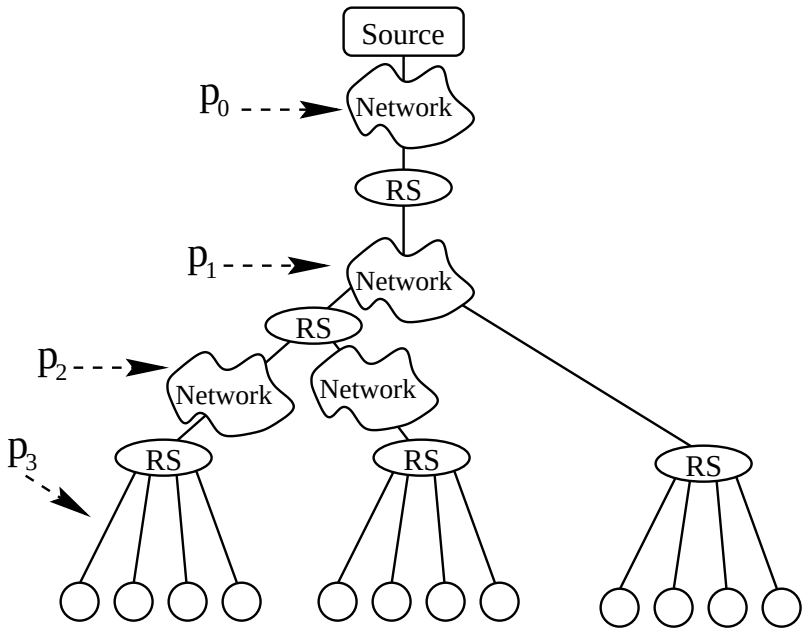

Figure 8: A hierarchy of repair servers.

Finally, it is important to note that among APES protocols, the GRSR protocol requires no FEC processing at repair servers, and the BRSR protocol requires no additional FEC processing at repair servers beyond what is required by protocol SDBR. Protocol SDBR also performs decode processing at the repair server. Thus, *the ordering of the protocols from least to most in terms of their FEC processing requirements is GRSR, BRSR, and SDBR.*

B.4 Receiver Feedback

We briefly consider how receivers convey to the repair server the repair packets that they desire to have transmitted. Figure 2 in Section 4 shows that if a receiver that loses m source packets requires the reliable transmission of repairs $k+1, \cdots, k+m$, then, for network conditions that are seen in practice, the bandwidth utilized is close to what can be achieved optimally if each repair packet transmitted by the repair server was unique (i.e., as in the SDBR protocol).

We propose that a receiver's initial NAK for each block indicate the number of repair packets, m, that it requires. The repair sends repair packets $k+1, \cdots, k+m'$, where m' is the maximum number of repairs requested by any receiver. Receivers await feedback, and make use of any distinct repair they receive. For instance, a receiver that loses $m < m'$ packets can use repair packet $k+m'$ in place of any repair $k+1, \cdots, k+m$ that it fails to receive. In subsequent rounds, each receiver determines the additional number of repairs that it needs, and using a bit-vector, requests that number of repairs with minimum sequence numbers that it has not yet received.

C Expected decoding processing at receiver

We now consider a network scenario where each receiver is serviced by a hierarchy of repair servers. We say that a repair server or receiver has a height of i if there are $i-1$ repair servers on its upstream path to the source. The source has height 0. We consider a single repair domain, and denote the probability of loss between the entity at height i and the entity at height $i+1$ as p_i. A sample hierarchical network is shown in Figure 8. The repair domain on the right is serviced by two repair servers, so receivers in this domain have a height of three, while the two domains on the left are serviced by three repair servers, so receivers in these domains heights of four. Loss rates are indicated for the leftmost domain.

It has been observed in [11, 3] that the decoding time of a decoder is roughly proportional to kd, where d is the number of packets that must be retrieved via decoding, and k is the block size used to encode the packets. We assume that the same block size is used in the encoding at each level of the hierarchy, and thus each repair server requires exactly k packets per block before it completes encoding. Let d_i be a random variable equal to the number of packets that were lost by a repair server at height $i+1$ from the transmission of k packets by the upstream repair server at height i. This repair server must obtain d_i repairs from its upstream repair server.

A receiver at height h must perform h recursive decodes to obtain the original data packets sent by the source. The ith decoding iteration (for the initial iteration, $i = 1$) begins with a set of k packets equivalent to those that the entity at height $h - i + 1$, takes an amount of time proportional to kd_{h-i}, and produces the k packets that were initially transmitted by the entity at height $h - i$. After the hth decoding iteration, the original source packets are retrieved. The expression for the total decoding time, $\mathcal{D}$, is

$$\mathcal{D} = k \sum_{i=0}^{h-1} d_i. \tag{19}$$

Since the loss rate between the entity at height i and height $i + 1$ is p_i, it follows that $E[d_i] = c_d k p_i$, where c_d is a constant associated with the decoder. Thus,

$$E[\mathcal{D}] = c_d k^2 \sum_{i=0}^{h-1} p_i. \tag{20}$$

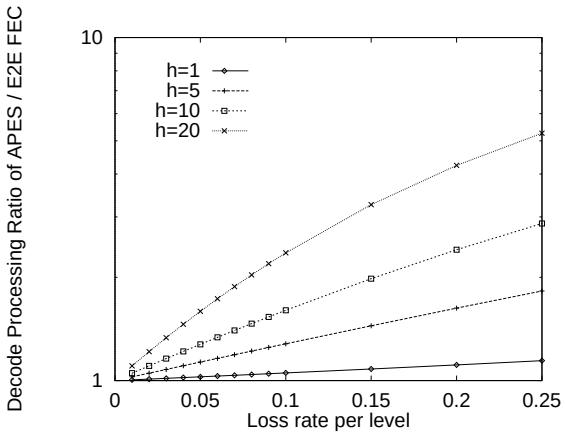

Figure 9: The ratio of decode processing between a receiver in a network that uses Protocol BRSR with that required by GRSR or of an end-to-end FEC protocol. The increase in processing is due to recursive decodes.

Protocol SDBR would only require a decoding time equal to $c_d k^2 p_{h-1}$, but each repair server would have to decode incoming repairs, and would not be able to do so until receiving all k packets for a block. If repair servers are not used, but FEC is sent directly from the source and the links maintain identical loss rates, then the probability of losing a packet is $1 - \prod_{i=0}^{h-1}(1 - p_i)$, and so the expected amount of decoding processing is $k^2(1 - \prod_{i=0}^{h-1}(1 - p_i))$. The ratio of expected decode processing at a receiver between Protocol BRSR and Protocol GRSR is given in Figure 9. Note that since all repairs are generated at the source for the GRSR protocol, the receiver's decoding processing is equal to what is required for an end-to-end, FEC based protocol.

The various plots are for receivers at differing heights, the x-axis represents the loss rate at each level of the hierarchy, (i.e. $p_i = p_j = p$ for all i, j). The end-to-end loss rate between source and receiver is $1 - (1 - p)^h$. We see that for values of p up to .1 and hierarchies containing heights as high as 10, the decoding cost in the Protocols BRSR is less than double that of the end-to-end FEC protocol.

When the loss rates on all links are identical, then the decoding ratio between Protocols BRSR and SDBR equals $h - 1$, where h is the height of the receivers within the domain being considered. This ratio is constant regardless of the loss rate.

D Avoiding slow-path processing of data

In order to perform decoding of FEC repairs to retrieve lost data, receivers must know precisely the order in which source packets were applied at the encoder to produce repairs at the repair server. For the BRSR protocol, this information was communicated via the repair server sequence number, which gives the order in which a packet was applied in the encoding process. However, this would require that source packets pass through the repair server before being forwarded to receivers downstream. We now present an alternative which is preferable in that source packets can be sent directly to receivers. This allows the source packets to be processed at each router on the fast path.

Each packet contains an additional bit which is set only in those packets which are source packets (i.e., the bit on every packet sent by the sender, and is unset on every packet sent by a repair server). Repair packets do not set the bit, and are sequenced starting from one, instead of from $k + 1$. For each block, an additional packet, which we call the *sequence order packet*, must be reliably transmitted to all downstream receivers. This packet contains an ordered list of the k sequence numbers of the source packets in the order that they were fed into the repair server's encoder. This information is necessary in order for the receiver to perform decoding.

A receiver can commence decoding once it any combination of k source and repair packets for a block, plus the block's sequence order packet. Note that the receiver is able to NAK for repairs prior to obtaining the sequence order packet. It simply cannot perform decoding. Thus, the buffer requirements at the repair server are not affected. However, there might be an additional latency in decoding if there is difficulty in obtaining the sequence order packet.

As long as the block size, k, is less than 256, the list in the sequence order packet consists of at most k bytes. Aside from the cost of a header, the bandwidth cost to deliver a sequence order packet is no more than if a sequence number were prepended to each source packet, as was done in our initial formulation of the protocol. Furthermore, this approach handles reordering of source packets.

This approach can be extended in at least two ways in an environment where there is a hierarchy of repair servers. Each repair server can reliably transmit its own sequence order packet to its set of downstream receivers. Alternatively, it can take the sequence order packet sent by its upstream repair server, and prepend its sequence order information to the packet and forward it downstream.