Parity-Based Loss Recovery for Reliable Multicast Transmission

Jörg Nonnenmacher, Ernst W. Biersack, and Don Towsley

Appeared in: IEEE/ACM Transactions on Networking, 6(4):349-361, August 1998

***Abstract*— We investigate how FEC (Forward Error Correction) can be combined with ARQ (Automatic Repeat Request) to achieve scalable reliable multicast transmission. We consider the two scenarios where FEC is introduced as a transparent layer underneath a reliable multicast layer that uses ARQ, and where FEC and ARQ are both integrated into a single layer that uses the retransmission of *parity* data to recover from the loss of original data packets.**

To evaluate the performance improvements due to FEC, we consider different loss rates and different types of loss behaviors (spatially or temporally correlated loss, homogeneous or heterogeneous loss) for up to 10^6 receivers. Our results show that introducing FEC as a transparent layer below ARQ can improve multicast transmission efficiency and scalability. However there are substantial additional improvements when FEC and ARQ are integrated.

1 Introduction

The deployment of the MBONE made multi-point communication across the Internet feasible and fostered the development of applications for video and audio distribution or collaborative work such as vic, vat, ivs, or wb. Most of these applications require real-time delivery and were built to tolerate loss or to perform partial (e.g. ivs) or complete loss recovery (e.g. wb) at the application level.

There is, however, a growing number of applications that could benefit from a reliable multicast transport service. To recover from loss, two well known techniques exist: ARQ (Automatic Repeat Request), which retransmits the lost data and FEC (Forward Error Correction) which transmits redundant data, called **parity** data along with the original data. With FEC, if the amount of original data lost is not more than the amount of parity data received, the parity data can be used to reconstruct the lost original data. Coding theory [1] distinguishes between two types of errors: (i) **corruption** of data, where bits are corrupted and (ii) **erasure** of data, where the whole packet is lost. While FEC is able to recover from both types of errors, we use FEC in this paper only to recover from erasures.

Jörg Nonnenmacher and Ernst W. Biersack are with Institut Eurecom, Sophia-Antipolis, France ({nonnen,erbi}@eurecom.fr). Eurecom's research is partially supported by its industrial partners: Ascom, Cegetel, France Telecom, Hitachi, IBM France, Motorola, Swisscom, Texas Instruments, and Thomson CSF.

Don Towsley is with the Dept. of Computer Science, Univ. of Massachusetts, USA (towsley@cs.umass.edu). The work of D. Towsley was supported in part by INRIA, by the Defense Advanced Research Project Agency under contract F19628-95–C-0146 and by the National Science Foundation under grants NCR-95-08274 and CDA-95-02639. Any opinions, findings, and conclusions or recommendations expressed in this paper are those of the authors and do not necessarily reflect the views of the National Science Foundation, INRIA, or the Defense Advanced Research Project Agency.

FEC by itself cannot provide full reliability. However, when coupled with ARQ, FEC can be used to produce inherently scalable reliable multicast transport protocols. If introduced as a separate layer beneath the ARQ layer, it has the effect of reducing the packet loss probability and thus reducing the number of packet retransmissions and network bandwidth requirements. If integrated with ARQ, then FEC has a very high repair efficiency and, therefore, substantially reduces the network bandwidth requirements of an application requiring reliable multicast data transport. Integrated FEC/ARQ schemes are also referred to in the literature as **hybrid ARQ** [2].

In this paper, we study the effectiveness of both approaches of combining FEC with ARQ. We find that a layered approach makes more efficient use of network resources than an approach based solely on ARQ except in conditions where losses are temporally very bursty. When integrated with ARQ, FEC provides a substantial reduction in the usage of network resources, even when losses are temporally correlated. Moreover, increased processing efficiency is achieved by the sender and receivers, even when FEC is implemented in software.

There is a large body of literature on the subject of reliable multicast - some of which focus on the use of FEC. An early paper by Metzner [3] studied the use of hybrid ARQ for reliable multi-point transmission in the context of a block data transfer with independent and homogeneous loss. Metzner suggested the use of Reed Solomon codes for parity computation and quantified the benefits of such a scheme in terms of the mean number of parity packets transmitted in the first retransmission round for a small number (up to 50) of receivers. Recently, Huitema [4] studied the benefits of FEC when used as a layer underneath the reliable multicast layer for the case of independent and homogeneous loss.

Most other reliable multicast protocols use only ARQ to assure reliability. In order to achieve scalability and avoid feedback implosion these protocols use slotting and damping [5, 6] or introduce a hierarchy [7, 8, 9]. Both approaches are not without disadvantages:

- Slotting and damping requires a careful choice/ estimation of parameters.
- Introducing a hierarchy poses the problem of selecting designated intermediate nodes that are required to have special functionalities. Also special care must be taken to cope with the failure of a designated node.

Coupled with these approaches, however, FEC can increase

efficiency and scalability.

The rest of the paper is organized as follows. Section 2 provides a brief overview of FEC. Section 3 compares the two approaches on how to combine FEC and ARQ in a reliable multicast protocol stack and evaluates their performance in the context of homogeneous and heterogeneous populations of receivers. Section 4 considers the performance of the two approaches in the presence of spatially and temporally correlated losses. As the focus of Sections 3 and 4 will be on network bandwidth requirements, Section 5 will focus on the end-host processing requirements by comparing the performance of two generic reliable multicast protocols: one with and one without FEC. Conclusions are drawn in Section 6.

2 How FEC works

2.1 Theory

The use of Forward Error Correction (FEC), as an alternative or complement to ARQ for reliable multicast transmission, has been investigated by different authors [3, 10, 11, 4, 12]. FEC involves the transmission of original data along with additional redundant data which can be used to reconstruct the original data if some of it is lost.

A Reed Solomon Erasure correcting code (RSE code), such as the one described by McAuley [13], is used to generate the redundant data. Suppose we have k data packets $d_1, d_2, \ldots, d_k$ each of which is P bits long. The RSE encoder takes $d_1, \ldots, d_k$ and produces **parities** $p_1, \ldots, p_{n-k}$ each P bits long. We also use the parameter h to denote the number $n-k$ of parities.

For the purpose of coding, we consider the vector $\vec{d} = [d_1, \ldots, d_k]$ of data packets as elements of the Galois field $GF(2^P)$ [1]. Given a primitive element α of $GF(2^P)$, a (n,k) matrix $G' = (g_{i,j})$ with elements in $GF(2^P)$ is defined:

$$g_{i,j} = \alpha^{i \cdot j}, 0 \leq i \leq 2^P - 2, 0 \leq j \leq k-1 \qquad (1)$$

The basic RSE coder can produce up to $n = 2^P - 1$ FEC packets as components of $\vec{y'} = [y'_1, y'_2, \ldots, y'_n]$:

$$\vec{y'} = G' \vec{d}^T \qquad (2)$$

The matrix G' has the property that *any* k out of the $2^P - 1$ row vectors are linearly independent. Therefore, at the RSE **decoder** any k components of $\vec{y'}$ are sufficient to uniquely specify $d_1, d_2, \ldots, d_k$.

This basic RSE scheme is not a systematic code, i.e., the data packets $d_1, d_2, \ldots, d_k$ are not part of $\vec{y'}$. As a consequence, the RSE decoder must *always* solve k simultaneous linear equations to retrieve the data packets $d_1, d_2, \ldots, d_k$ from k components of $\vec{y'}$.

Fortunately, there is a simple solution to avoid this decoding complexity. Prior to coding, gaussian elimination on the matrix G' is used, to turn the first k row vectors of G' into a (k,k) identity matrix. Using this G, coding $\vec{y} = G\vec{d}^T$ results in the first k components of $\vec{y}$ being copies of $d_1, d_2, \ldots, d_k$. The other components of $\vec{y}$ are the parities: $p_i = y_{k+i}$ for $i \in \{1, \ldots, n-k\}$.

The RSE **decoder** at the receiver side can reconstruct the data packets $d_1, \ldots, d_k$, whenever it has received *any* k out of the n packets $d_1, \ldots, d_k, p_1, \ldots, p_{n-k}$. The k data packets will also be referred to as a **transmission group**, or **TG**. The n packets $d_1, \ldots, d_k, p_1, \ldots, p_{n-k}$ will be referred to as a **FEC block**. Sending the original data as the first k packets of the FEC block simplifies decoding:

- If all k data packets are received, no decoding at all is required at the receiver.
- If $l < n-k$ out of the k data packets are lost, the decoding overhead is proportional to l.

There are multiple benefits using parity packets for loss recovery instead of retransmitting the lost original data packets:

- **Improved transmission efficiency:**
  A single parity packet can be used to repair the loss of *any* one of the n data packets. This means that a *single parity packet* can repair the loss of *different data packets at different receivers*. Hence, FEC is particularly well suited for a multicast scenario.
- **Improved scalability in terms of group size:**
  An ARQ scheme which retransmits the original packets requires the sender to know the sequence numbers of the lost packets. By using parity packets for loss repair, the sender only needs to know the maximum number of packets lost by any receiver within a TG but not their sequence numbers. Feedback is reduced from per–packet feedback to per–TG feedback.
- **Reduction of unnecessary receptions:**
  Multicast retransmissions for loss recovery result in unnecessary receptions at all receivers that do not need the retransmission. Such unnecessary receptions waste processing capacity. As we will show later, the number of unnecessary receptions is significantly reduced with parity transmission.

2.2 Practical Aspects

The size of a packet, P, will typically be on the order of several hundred bits (ATM cell) to several thousand bits (IP datagram). RSE coders that operate on symbols of that size are difficult to implement. The hardware architecture for the RSE coder proposed by McAuley [13] uses a symbol size m with $m = 8$ or $m = 32$. The software (SW) implementation of the coder by Rizzo [14] uses $m = 8$ and can therefore use a fast table lookup for multiplication and division.

In the case of $P > m$, we need to choose $P = S \cdot m$ where S is an integer. We then perform multiple parallel RSE encodings for each m-bit symbol in each data packet (see Figure 2 of [13]). For example, we perform RSE on the first m-bit symbol of each of the k data packets and obtain $n-k$ m-bit parity packets. We repeat this on the 2nd m bit symbol of the k data packets and so on.

The number of elements in a Galois field $GF(2^m)$ is lim-

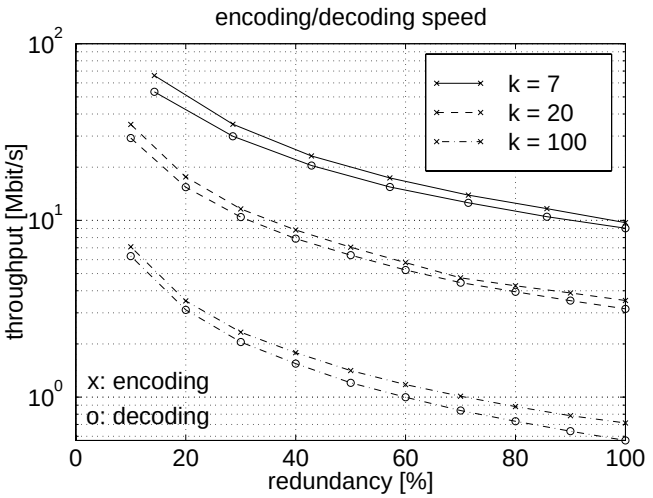

Figure 1: Coder and decoder throughput in Mbits/s with respect to the percentage of redundancy $h/k \cdot 100\%$ and the TG size k.

ited to 2^m elements. Therefore, the symbol size m must be picked sufficiently large such that $n < 2^m$. For our purposes, $m = 8$ will be sufficiently large.

In order to assess the encoding and decoding speed we measured the throughput of the software coder by Rizzo on a Pentium PC 133 running Linux. The amount of original data is split equally into k data packets of $P = 1$ KByte each. The amount of redundancy, given as the percentage $h/k \cdot 100\%$ of the original data, is produced (encoding) or used for reconstruction (decoding) together with the originals received.

Figure 1 shows the encoding and decoding throughput. The encoding throughput denotes the amount of original data processed per second, given that a percentage $h/k \cdot 100\%$ of parity data is produced from the original data. The decoding throughput denotes the amount of parity and original data processed per second to reconstruct a percentage of $h/k \cdot 100\%$ of original data that are lost.

The analysis of the coding and decoding algorithm in [14] is confirmed by our own measurements. We observe that:

- The encoding speed is inversely proportional to the amount of parity data produced.
- The decoding speed is inversely proportional to the amount of original data lost and reconstructed, i.e. the amount of parity data used for reconstruction.
- Both encoding and decoding throughput are inversely proportional to the TG size $k = 7, 20, 100$ (see eq. 16 and eq. 17 in the Appendix).
- The decoding throughput is slightly less than the encoding throughput.
- For a constant TG size, k, the throughput in Mbit/s of the encoder and decoder is insensitive to the packet size P.

If we look at the absolute encoding performance (Figure 1), we observe in the case of a TG size of $k = 7$, that we

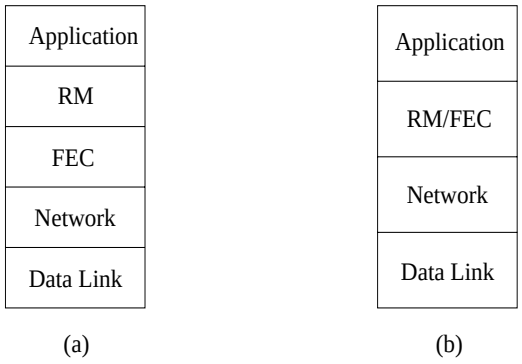

Figure 2: (a) Layered FEC. (b) Integrated FEC.

achieve a coding rate of 65 Mbit/s, when one parity packet is produced (14.3% redundancy). This means the first parity packet for a TG of size $k = 7$ is available after 126 μs. Such high performance and the fact that the transmission rates of many current multicast applications are typically less than 100 KByte/s, suggests that coding in software will not affect the packet sending rate and that loss recovery using parity data is feasible. This point will be further investigated in Section 5.1.

3 Placement of FEC in a Reliable Multicast Protocol Stack

There are a number of different ways that FEC can be introduced to a reliable multicast protocol stack. The simplest approach is to add a layer responsible for FEC between the network layer and the reliable multicast layer (RM), such that the RM layer sees a more reliable network. The second approach is to integrate FEC with reliable multicast and place it into a single layer. These approaches, henceforth referred to as **layered FEC** and **integrated FEC** are illustrated in Figure 2.

The advantage of layered FEC is the separation of FEC and the reliable multicast layer. Thus a designer can focus on the problem of reliable multicast without worrying about the intricacies of FEC. In addition, an FEC layer can be used by other applications that may benefit from a more reliable network. Huitema [4] has established the benefits of layered FEC in terms of reducing the network traffic. We will review this work in Section 3.1.

Given the simplicity of layered FEC and its potential performance benefits, it is reasonable to ask whether integrating FEC and reliable multicast can provide additional performance benefits that would outweigh the additional complexity required of such an approach. We shall examine this question in Section 3.2.

The emphasis of this section and Section 4 will be to compare the average number of transmissions required to transmit a packet reliably to all receivers using layered FEC, integrated FEC, and no FEC. The average number of transmissions is an important metric because it reflects directly the network bandwidth required to support reliable

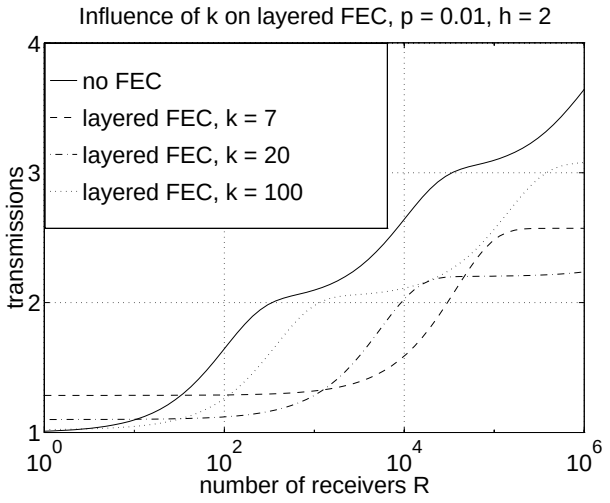

Figure 3: Non FEC versus layered FEC with $h = 2$ parity packets for different TG sizes $k = 7, 20, 100$ and loss probability $p = 10^{-2}$.

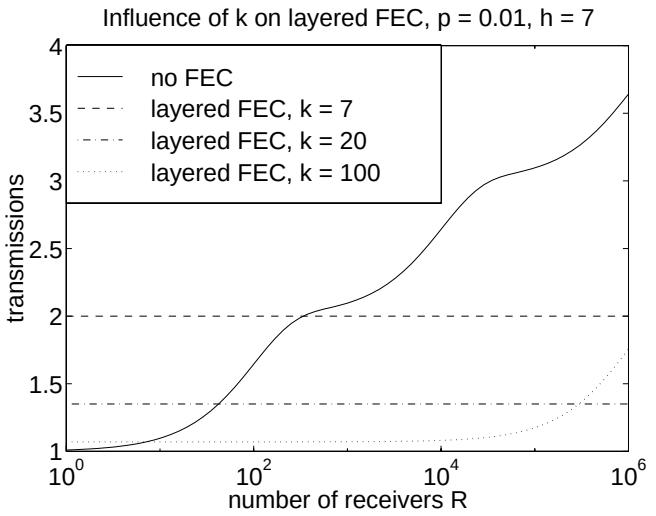

Figure 4: Non FEC versus layered FEC with $h = 7$ parity packets for different TG sizes $k = 7, 20, 100$ and loss probability $p = 10^{-2}$.

multicast. In addition, it is also correlated to the processing requirements at the end/hosts (which will be examined in Section 5). Last, although we do not examine the latency reduction benefits of FEC, we expect that a reduction in the required number of transmissions will often lead to a reduction in latency, especially for a multicast transmission of large amounts of data (bulk transfer).

Throughout this section we will consider a single sender with an infinite supply of packets sending to R receivers. We assume that packet losses occur as independent events, both spatially and temporally with probability p. We examine the effect that different loss probabilities have on the performance of reliable multicast in Section 3.3.

3.1 Layered FEC

Consider layered FEC based on the RSE codes described in the previous section. At the sender, the RM layer passes the packets to be sent to the FEC layer. The FEC layer constructs and sends a FEC block: it takes k original packets, produces h parities and sends all of the packets to the receiving FEC layer. Whenever the FEC layer receives an original data packet, it passes it to the RM layer and keeps a copy for decoding purposes in case of loss. Whenever the FEC layer receives at least k out of $k + h$ packets, all of the lost original packets are reconstructed and delivered to the RM layer. If fewer than $k+h$ packets are received, the lost original packets cannot be reconstructed and the FEC layer discards the received parity packets. The sending RM layer then retransmits the lost originals as part of a new FEC block.

Let $q(k, n, p)$ denote the probability that the RM receiver does not receive a random data packet from the TG sent by the RM sender. In particular, a packet from a TG is lost at the RM receiver if it is lost by the FEC receiver and if more than $h - 1 = n - k - 1$ of the other $n - 1$ packets from the FEC block are lost. This was first proposed and studied in [4]. Therefore the packet loss probability at the RM receiver is for $1 \leq k \leq n$ given by

$$q(k, n, p) = p \left(1 - \sum_{j=0}^{n-k-1} \binom{n-1}{j} p^j (1-p)^{n-j-1} \right) \quad (3)$$

Let M' denote the number of times an arbitrary data packet is transmitted before all RM receivers have received it. The probability that fewer than $i + 1$ data packet transmissions are required by a single RM receiver is $1 - q(k, n, p)^i, i = 0, 1, \ldots$. The cumulative distribution of M' is therefore:

$$P(M' \leq i) = (1 - q(k, n, p)^i)^R, i = 0, 1, \ldots$$

Let M be the equivalent of M' with the additional accounting of parity packets added at the FEC layer. This quantity is difficult to define precisely; however, its average is given by

$$\begin{aligned} E[M] &= n/k \cdot E[M'] \\ &= n/k \cdot \sum_{i=0}^{\infty} (1 - P(M' \leq i)) \end{aligned} \quad (4)$$

Figures 3 and 4 illustrate the typical behavior of layered FEC for different size transmission groups, $k = 7, 20, 100$, when the numbers of parity packets are $h = 2$ and $h = 7$. First, we observe a decrease in the expected number of transmissions when FEC is introduced and the receiver population is large; an observation first made in [4]. Second, FEC introduces a constant overhead of h/k, which results in $E[M] \approx n/k$ for a small number of receivers: this overhead results in worse performance than a non FEC protocol. Once the number of receivers is sufficiently large, the FEC overhead is amortized by the usage of FEC packets to repair different losses at different receivers and the

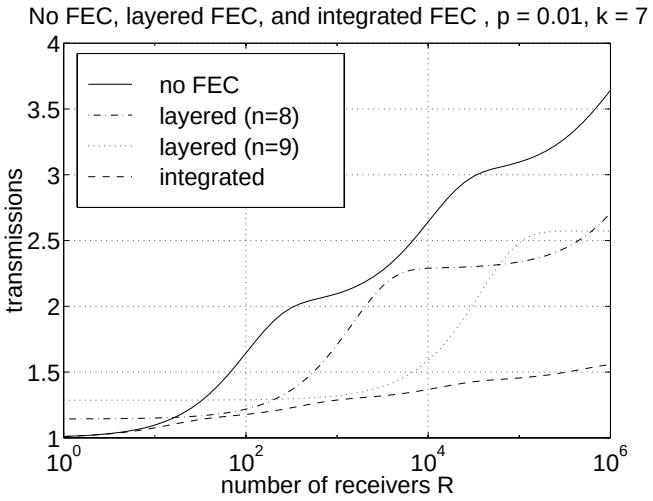

Figure 5: Comparison of non FEC, layered FEC ($n = 8, 9$), and integrated FEC ($n = \infty$) for $k = 7$ and $p = 10^{-2}$.

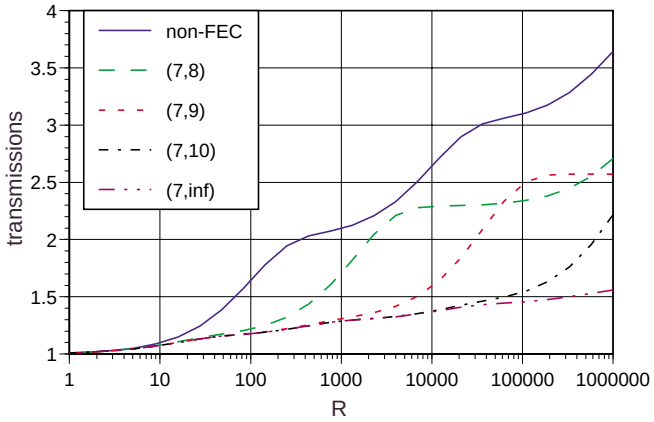

Figure 6: Integrated FEC with $k = 7$ and $p = 10^{-2}$ for different values of $n = 8, 9, 10, \infty$.

introduction of FEC for large numbers of receivers results in a better performance than a non FEC approach.

Figures 3 and 4 show the difficulty of choosing the right parameters h and k for layered FEC. For example, a TG of size $k = 100$ with $h = 2$ parity packets exhibits worse performance than TGs of sizes $k = 7$ and $k = 20$ with the same number of parity packets. On the other hand a TG of size $k = 100$ coupled with $h = 7$ parity packets performs better than TGs of sizes $k = 7$ and $k = 20$ for receiver populations in the 1 to 200,000 range. We have observed similar behavior for a wide range of packet loss probabilities.

The mean number $E[M]$ of transmissions is logarithmically increasing with the number of receivers. The curve for $E[M]$ has a periodicity of $-\log(q(k, n, p))$ and a step like behavior with increments of n/k. As the loss probability $q(k, n, p)$ becomes smaller, the period becomes longer and the steps flatter. For a detailed analysis of the behavior of $E[M]$ see [15].

3.2 Integrated FEC

We now turn our attention to integrated FEC where the RM layer uses FEC to enhance its performance.

There are many ways that FEC can be included within the RM layer. We will propose and evaluate one such protocol in Section 5.1. In order to assess the potential benefits without becoming involved in a detailed analysis, we will study the following generic protocol.

- The sender sends a TG along with $a \leq h$ parity packets from the associated FEC block.
- If there are a or fewer missing packets among the $k+a$ packets sent, all packets within the TG can be recovered. Each time that a receiver detects a missing packet beyond a, it requests a new parity packet from the sender until it has received a sufficient number of packets (k) out of the n packets in the FEC block to complete the decoding of all k packets.
- The sender multicasts parity packets in response to requests until all parity packets associated with the TG have been used up. At that time, packets requiring retransmission are placed into a new TG.

We first derive an unachievable lower bound to the expected number of packet transmissions required to transmit an arbitrary packet to all receivers. This corresponds to the performance of the above protocol when $n = \infty$. Let L_r denote the number of additional packet transmissions required by a random receiver. The distribution of L_r is

$$P(L_r = 0) = \sum_{j=0}^{a} \binom{k+a}{j} p^j (1-p)^{k+a-j},$$

$$P(L_r = m) = \binom{k+a+m-1}{k-1} p^{m+a} (1-p)^k, \quad m = 1, \ldots$$

Let L denote the maximum number of additional packets that are transmitted. Its cumulative distribution is given by

$$P(L \leq m) = [P(L_r \leq m)]^R, \quad m = 0, 1, \ldots \tag{5}$$

where

$$P(L_r \leq m) = \sum_{i=0}^{m} P(L_r = i), \quad m = 0, 1, \ldots$$

The mean number of additional transmissions is

$$E[L] = \sum_{m=0}^{\infty} (1 - P(L \leq m)). \tag{6}$$

Finally, defining M as before, the mean number of transmissions per arbitrary packet is

$$E[M] = (E[L] + k + a)/k. \tag{7}$$

The protocol analyzed here sends parities pro-actively when $a > 0$. Pro-actively transmitted parities reduce the feedback and may benefit applications with time constraints.

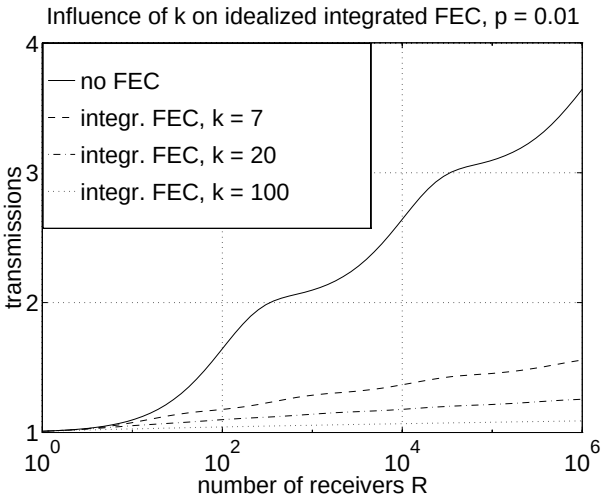

Figure 7: Influence of number of receivers on integrated FEC with different TG sizes $k = 7, 20, 100$ and packet loss probability $p = 10^{-2}$.

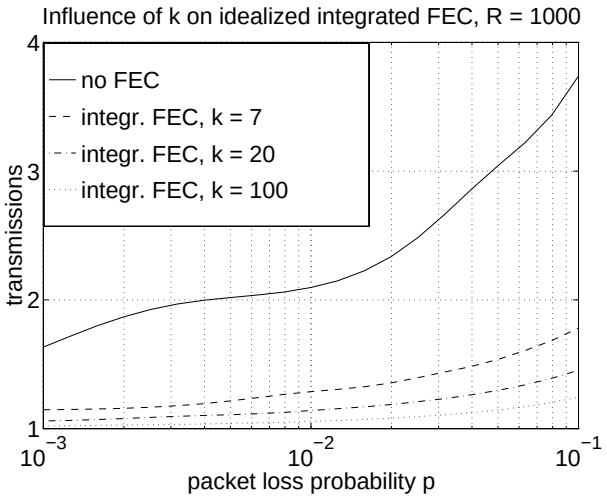

Figure 8: Influence of loss probability p on integrated FEC with different TG sizes $k = 7, 20, 100$ and $R = 1000$ receivers.

On the other hand the previous section on layered FEC shows that transmitting unused parities lead to worse performance than with no FEC, when the number of receivers is small (compare Figures 3 and 4). In the remainder of the paper we no longer consider pro-active sending of parities and $a = 0$.

Next we derive an expression for $E[M]$ in the case that $n < \infty$. Let B denote the number of times that an arbitrary packet was transmitted, i.e., the number of blocks transmitted that included it. Note that the transmission of the first $B-1$ groups requires the transmission of n packets each, just like layered FEC. On the other hand, the number of packets transmitted as part of the last group is a random variable whose distribution is identical to that of L given that $L \leq n$. Given this, the expected number of transmissions is

$$\begin{aligned} E[M] &= ((E[B]-1)n + E[L \mid L \leq n])/n \\ &= \frac{n}{k}\left(\sum_{i=1}^{\infty} 1 - (1 - q(k,n,p)^i)^R\right) \\ &\quad + \frac{1}{k}\sum_{m=0}^{n}(1 - P(L \leq m \mid L \leq n)) \end{aligned}$$

where $q(k,n,p)$ is computed using expression (3) and $P(L \leq m \mid L \leq n)$ is the properly conditioned version of $P(L \leq m)$ given in (5).

Figure 5 compares the expected number of transmissions per correctly received packet under layered FEC to a lower bound under integrated FEC for a TG size of 7 and loss probability $p = 10^{-2}$. We observe that integrated FEC has the potential for a large performance improvement over layered FEC. In Figure 6 we examine integrated FEC more closely to determine how many parity packets are needed to achieve a performance close to that of the lower bound. We observe that in the case of a TG size of $k = 7$, $h = 3$ parity packets suffice to attain the lower bound for receiver populations up to 100,000. Henceforth, we will use the lower bound when comparing integrated FEC to other approaches.

Last, Figure 7 shows the influence of the TG size k on the performance of integrated FEC. Increasing the TG size reduces the number of transmissions for integrated FEC to nearly one, even for a large number of receivers. From Figure 8 we observe that this behavior is fairly insensitive to the loss probability; a large increase in the loss probability has little effect on the performance of integrated FEC.

3.3 Heterogeneous Receivers

We end this section with a discussion of how our observations change in the presence of heterogeneous receivers, i.e., receivers with different loss probabilities. Let $p(r)$ denote the probability that receiver $r = 1, \ldots, R$ loses a packet. If we continue to assume that losses are spatially and temporally independent, then for layered FEC

$$E[M] = \sum_{i=0}^{\infty}(1 - \prod_{r=1}^{R}(1 - q(k,n,p(r))^i))n/k \qquad (8)$$

In the case of integrated FEC, $E[M]$ is given by equation (7) with

$$P(L \leq m) = \prod_{r=1}^{R} P(L_r \leq m) \qquad (9)$$

Here $P(L_r \leq m)$ is calculated using $p(r)$ in place of p. We consider a population consisting of two classes of heterogeneous receivers; $R \cdot (1-\nu)$ receivers with packet loss probability $p(r) = 10^{-2}$ and $R \cdot \nu$ high loss receivers with packet loss probability $p(r) = 0.25$. This allows us to vary the fraction ν of high loss receivers among all receivers.

We investigate the degradation in performance (increase in $E[M]$) as the number of high loss receivers increases. In particular, we take the percentage, $\nu \times 100\%$, of high loss receivers to be $1\%, 5\%$, and 25% of the whole group.

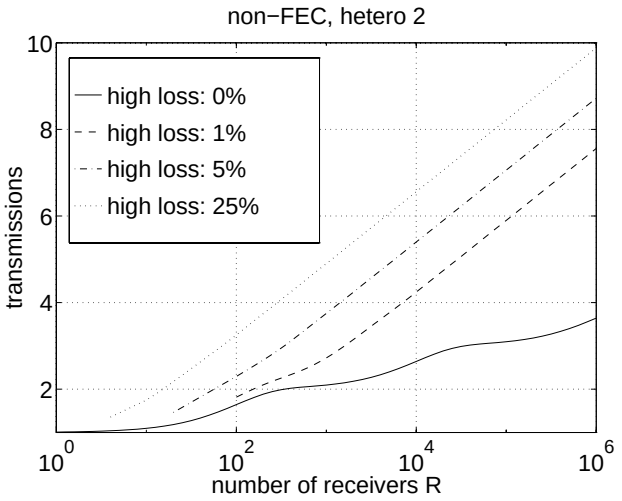

Figure 9: Reliable multicast without FEC for different fractions of high loss receivers.

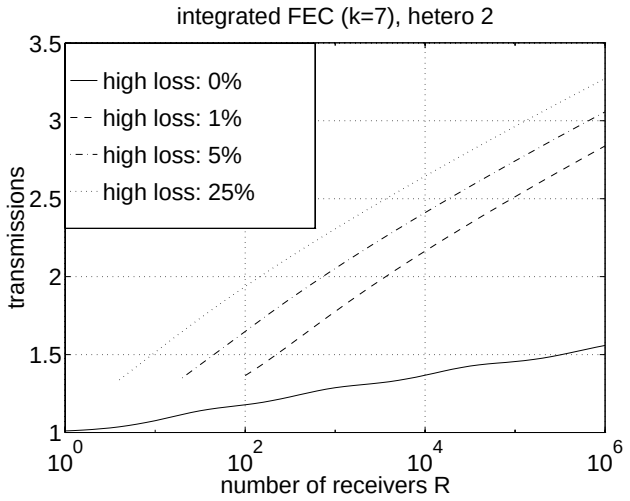

Figure 10: Reliable multicast with integrated FEC for TG size $k = 7$ and different fractions of high loss receivers.

The results for reliable multicast without FEC (Figure 9) and with integrated FEC (Figure 10) are similar. It can be seen that the influence of the high loss receivers increases with the number of receivers. For a group of one million receivers, the presence of 10,000 high loss receivers (1% of the population) is sufficient to double the expected number of transmissions (Figures 9 and 10). On the other hand, the presence of one high loss receiver in a population of $R = 100$ has much less effect on the expected number of transmissions. Comparing Figures 9 and 10 we observe that the presence of high loss receivers has a greater effect in the case of integrated FEC than no FEC.

We further observe that the performance is almost solely determined by the receivers with high loss rate. For integrated FEC and no FEC we observe that the increase of high loss receivers from 1% over 5% to 25% results in essentially the same curves with a linear translation in the number R of receivers on a logarithmic scale.

For real multicast groups the percentage of high loss receivers is in most cases determined by the position of a high loss router in the multicast tree and the number of receivers that experience the high loss of this router. In this case loss is spatially correlated as the receivers downstream will be equally affected by a loss. In the next section we will examine the influence of spatial correlation on reliable multicast with FEC.

4 The Effect of correlated loss on the FEC/ARQ performance

Until now, our focus has been on a scenario where losses are spatially and temporally uncorrelated. In this section we relax each of these assumptions in an attempt to understand whether and how correlated losses may affect our conclusions. Section 4.1 focuses on spatial loss correlation and Section 4.2 focuses on temporal loss correlation.

4.1 Shared Loss

Consider a sender and R receivers connected by an arbitrary multicast tree. Until now we have assumed that all losses occur as independent events at the receivers. In a real situation, however, there will be loss within the tree which will be shared by more than one receiver. In this section we explore whether and how the presence of such losses affects the conclusions which we have drawn from the independent loss model. In [16] the sharing of loss was analyzed for multicast trees built by different multicast routing algorithms. The authors conclude that the loss sharing in multicast trees is modeled well by a full binary tree (FBT). In order to investigate the impact of shared loss we consider a FBT of height d, where the source is the root of the tree and the receivers are the leaves. We compare FEC for:

- **Shared loss**: Losses occur as independent events at each node (including source and leaves) with probability p_n. Here p_n is chosen such that the loss probability at each receiver is p:

$$p = 1 - (1 - p_n)^{(d+1)}.$$

- **Independent loss**: Only the receivers lose packets, each receiver independently with probability p. Other nodes of the multicast tree do not lose packets at all.

Note that each receiver experiences the same loss probability in both models and that there is no temporal loss correlation.

The expected number of transmissions required to correctly transmit a packet reliably over a FBT was first derived in [17]. Because the calculation of this quantity is computationally intensive for large numbers of receivers, we use simulation to investigate the impact of shared loss for numbers of receivers: $R = 2^d$, $d = 0, \ldots, 17$. The packet loss probability is $p = 10^{-2}$, FEC was evaluated under the assumption that transmission groups were of size $k = 7$ with one transmitted parity packet ($h = 1$) in the case of

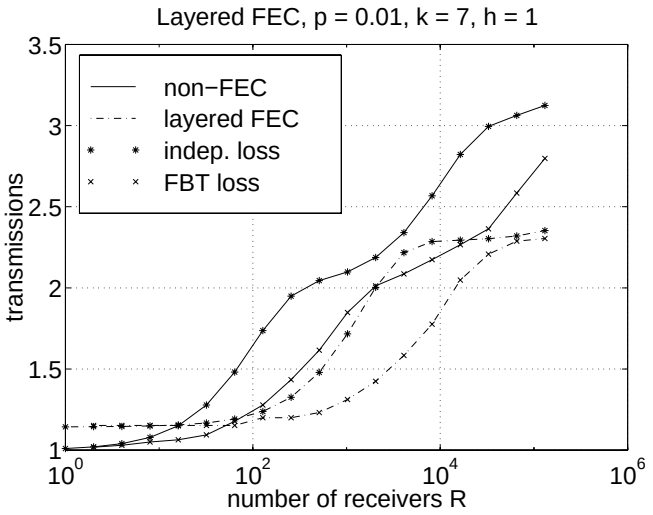

Figure 11: Layered FEC with $k = 7$ and $h = 1$ versus non-FEC for Independent and for FBT Shared Loss with loss probability $p = 0.01$.

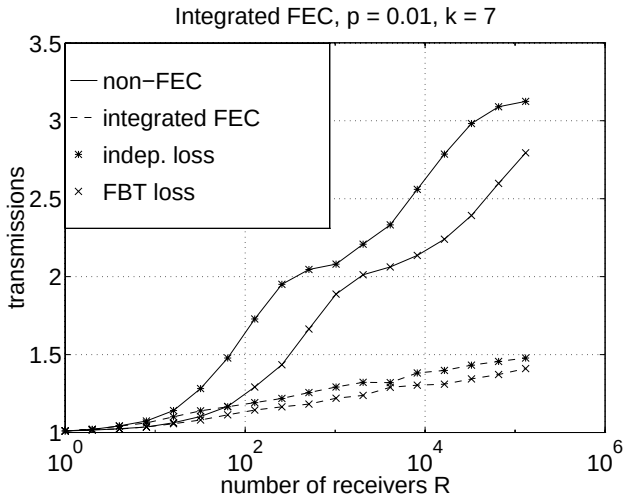

Figure 12: Integrated FEC with $k = 7$ versus non-FEC for Independent and for FBT Shared Loss with loss probability $p = 0.01$.

layered FEC.

The impact of shared loss on FEC is shown in Figure 11 for layered FEC and in Figure 12 for integrated FEC. First, we observe that the mean number of transmissions is lower (often substantially) when losses are shared than when they are independent. Second, we observe that our observations drawn from the independent loss model in the previous section continue to hold. However, receiver group sizes need to be larger before the benefits of layered FEC appear. This is because a parity does not exhibit the same repair efficiency under shared losses as it does in the presence of independent losses and may be transmitted needlessly. Figure 11 shows that the overhead of transmitted parities with layered FEC is amortized by the repair efficiency for a number of receivers $R > 60$ in the case of shared loss, whereas in the case of independent loss, layered FEC is already efficient for $R > 20$.

Another useful observation is that the curves for shared loss appear as translated versions of the independent loss curves and that the performance of a group of size R, regardless of whether FEC is used or not, can be determined by studying the behavior of a group of size $R_{indep} \leq R$ under the independent loss model. In fact, the extreme case is when all losses are shared by all receivers in which case the system can be modeled by a single receiver under the independent loss model. This carries the following important implication:

- Adaptive transport mechanisms that are based on measurements of receiver loss rates will overestimate the number of transmissions of reliable multicast if they model losses as independent events. Coupled with FEC this could lead to an overestimate of the amount of redundancy needed.

In summary, our results show that shared loss will result in a lower number of transmissions compared to independent loss for all recovery schemes and that the improvement of reliable multicast transmission due to FEC (compare non–FEC and FEC) is lower when losses are shared than when they are totally uncorrelated.

4.2 Burst Losses

In this section we reexamine the benefits of FEC when losses are bursty. In particular, we assume that packet losses are described by a two state continuous time Markov chain $\{X_t\}$ where $X_t \in \{0, 1\}$. A packet transferred at time t is lost if $X_t = 1$ and not lost if $X_t = 0$. The infinitesimal generator of this Markov chain is

$$\mathbf{Q} = \begin{bmatrix} -\mu_0 & \mu_0 \\ \mu_1 & -\mu_1 \end{bmatrix}$$

The stationary distribution associated with this chain is $\pi = (\pi_0, \pi_1)$ where $\pi_0 = \mu_1/(\mu_0 + \mu_1)$ and $\pi_1 = \mu_0/(\mu_0 + \mu_1)$. Let $p_{i,j}(t)$ denote the probability that the process is in state j at time $t + \tau$ given that it was in state i at time τ, $p_{i,j}(t) = P(X_{\tau+t} = j \,|\, X_\tau = i)$. These probabilities are given by [18, ch. 6].

$$p_{i,j}(t) = \begin{cases} \mu_1(1 - \exp(-\zeta t))/\zeta, & i = 1, j = 0, \\ \mu_0(1 - \exp(-\zeta t))/\zeta, & i = 0, j = 1, \\ (\mu_0 + \mu_1 \exp(-\zeta t))/\zeta, & i = 1, j = 1, \\ (\mu_1 + \mu_0 \exp(-\zeta t))/\zeta, & i = 0, j = 0 \end{cases}$$

for all $t > 0$ and $\zeta = (\mu_0 + \mu_1)$.

We now consider what effect this kind of loss process has on the expected number of packet transmissions required for each correctly received packet in the absence of FEC, with layered FEC, and with integrated FEC. In all cases, we assume that the loss processes are independent from receiver to receiver.

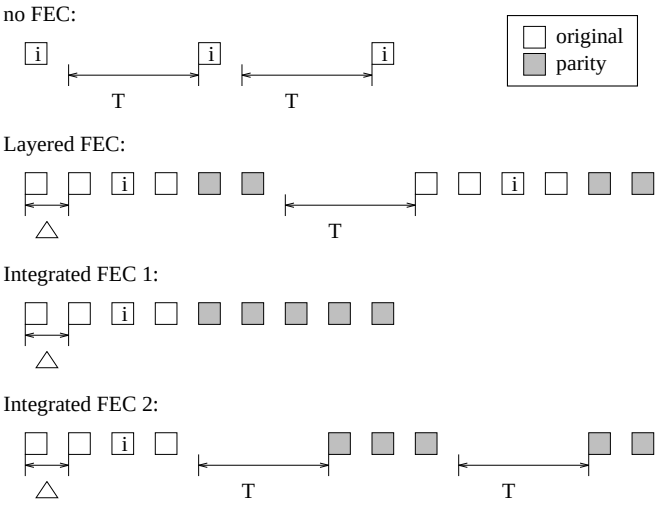

Figure 13: Timing considerations of the different approaches for data and parity retransmission.

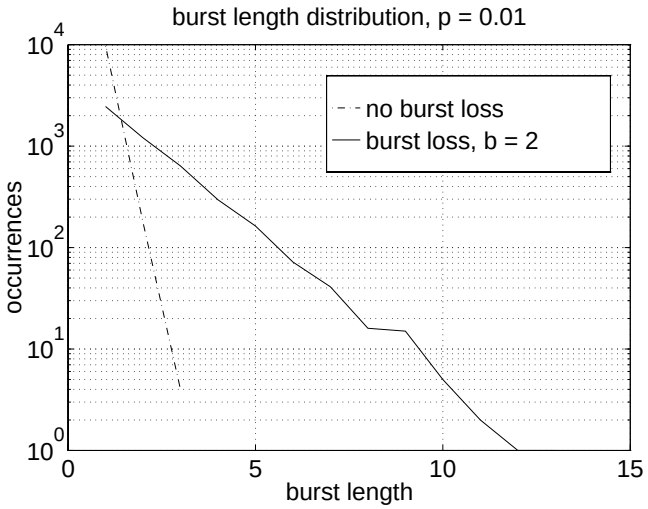

Figure 14: Distribution of number of consecutive losses at one receiver, for no burst and burst loss ($\bar{b} = 2$) for a packet loss probability of $p = 0.01$.

Let $\lambda = 1/\Delta$ be the packet transmission rate and $\bar{b}$ be the expected number of consecutively lost packets. Given the packet loss probability p, the average burst loss length $\bar{b}$ in packets and the sending rate λ, the parameters for the loss model are:

$$\begin{aligned} \mu_0 &= -p\lambda \log\left(1 - 1/\bar{b}\right) \\ \mu_1 &= \mu_0(1-p)/p \end{aligned}$$

When burst losses occur, the timing of the retransmissions influences the performance of loss recovery. To further investigate this point, we consider different cases as shown in Figure 13.

- **No FEC**: In the absence of FEC, we assume the time between successive transmissions of the same packet to be spaced by time $\Delta + T$.
- **Layered FEC**: For layered FEC, we assume the time between the sending of the last packet of a FEC block and the time of the sending of the first packet in the successive FEC block, containing the retransmission, to be spaced by time $\Delta + T$. We further assume that a packet keeps its place in the FEC block for retransmission.

For integrated FEC, the timing considerations depend on the protocol that implements loss recovery by parity transmissions. We distinguish between two cases:

- **Integrated FEC 1**: Parity packets are transmitted with the same rate $1/\Delta$ immediately following the original packets of the TG. When a receiver has received enough parity packets for the TG, it leaves the multicast group and therefore stops receiving packets. No feedback is needed for loss recovery and there is no unnecessary delivery and reception of parity packets, provided that the time needed to depart from the group is smaller than the packet inter-arrival time. We limit ourselves to the transmission of one TG, though several TGs can be delivered, staggered in time on different multicast channels.
- **Integrated FEC 2**: This protocol corresponds to a hybrid ARQ protocol, where receivers send NAKs indicating the number of missing packets. Feedback is sent after the transmission of the original packets, after the first retransmission of parities, etc.. Subsequently the sender transmits the maximum number of parity packets needed by any receiver.

Integrated FEC 2 is motivated by the fact that interleaving improves the performance of FEC in case of burst loss [1]. Interleaving allows the sender to spread the transmission of a FEC block over an interval that is longer than the burst-loss length. The benefit of interleaving is that burst loss is transformed into random loss. Integrated FEC 2 spaces parity transmissions out by intervals of length $\Delta+T$, whereas Integrated FEC 1 sends all parities just spaced by Δ. The term interleaving comes from the fact that packets from different TGs can be sent simultaneously in an interleaved manner. Note that during the gaps of duration T packets belonging to other TGs are sent.

First, we will analyze no FEC under the burst loss model. Let M_r denote the number of transmissions required for receiver r to receive an arbitrary packet. Its distribution is

$$P(M_r = m) = \begin{cases} \pi_0, & m = 1, \\ \pi_1 p_{1,1}(\Delta+T)^{m-2} p_{1,0}(\Delta+T), & m \geq 2 \end{cases}$$

Let M denote the total number of transmissions required to get the packet to all R receivers, It has cumulative distribution

$$P(M \leq m) = \left(1 - \pi_1 p_{1,1}^{m-1}(\Delta+T)\right)^R, \qquad m = 1, \ldots$$

The expectation of M, is given as

$$E[M] = \sum_{m=1}^{\infty} 1 - P(M \leq m)$$

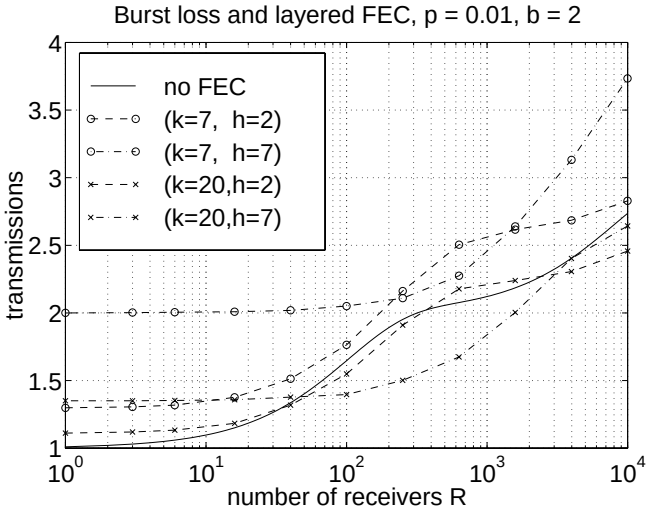

Figure 15: Comparison of reliable multicast without FEC and with layered FEC for small ($n = 7+2$, $n = 7+7$) and large FEC blocks ($n = 20+2$, $n = 20+7$), $p = 0.01$ and $\bar{b} = 2$.

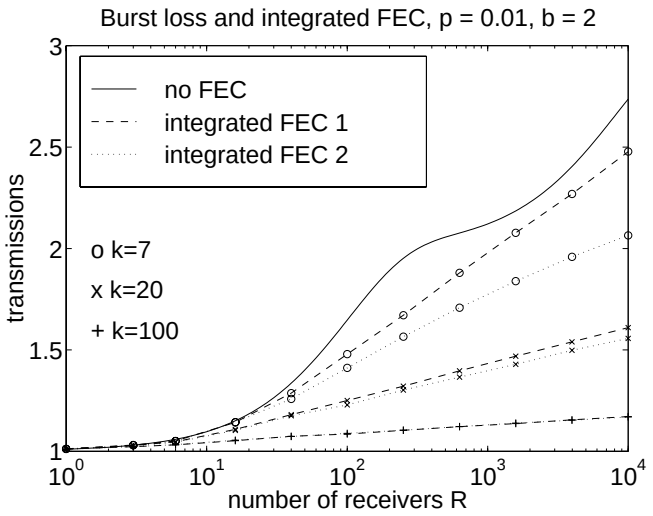

Figure 16: Comparison of integrated FEC 1 and integrated FEC 2 for TG sizes $k = 7, 20, 100$, $p = 0.01$ and $\bar{b} = 2$.

We use simulation to examine the impact of burst loss on the FEC schemes. We choose an average burst length of $\bar{b} = 2$, and $\Delta = 40$ms corresponding to a sending rate of $\lambda = 25$ packets/s as reported by Bolot [19] for a loaded IP path between INRIA (Sophia Antipolis, France) and UCL (London, UK). The packet loss probability is chosen to be $p = 0.01$, T is chosen to be 300 ms.

Figure 14 illustrates the burst length distribution at one receiver under the independent and burst loss models for these parameters. It can be seen that the tails of both distributions decrease linearly on a logarithmic scale.

We observe from Figure 15 that layered FEC performs worse than reliable multicast without FEC in the presence of burst losses when the TG consists of $k = 7$ packets. A larger FEC block size $n = 20 + 7$ permits layered FEC to perform slightly better than no FEC only if the number of receivers is large. However increasing the FEC block size is not always desirable since the FEC layer is no longer transparent to the RM layer due to high recovery latencies.

While large TG sizes are not desirable under layered FEC, it is reasonable to consider large TGs under integrated FEC. Figure 16 shows the performance of integrated FEC 1 and integrated FEC 2 for different TG sizes ($k = 7, 20, 100$). For a small TG size of $k = 7$, integrated FEC 1 and integrated FEC 2 outperform reliable multicast without FEC only slightly in the presence of burst loss. Integrated FEC 2 performs better than integrated FEC 1 for $k = 7$, since parity packets belonging to the same TG are spread out over time (see Figure 13) and are more likely to bridge a loss period. Figure 16 also shows that increasing the TG size from $k = 7$, to $k = 20$ and $k = 100$ significantly improves the performance of integrated FEC. Furthermore, there is little difference between integrated FEC 1 and integrated FEC 2 primarily due to the fact that the transmission of a TG is spread over a sufficiently long period of time to span a loss period such that subsequent parity packets are unlikely to be affected by it. This shows that a large TG size is sufficient to resist burst loss and that additional interleaving (integrated FEC 2) is not necessary.

5 End-host throughput of a hybrid ARQ protocol

In the previous sections we showed that integrating FEC with reliable multicast greatly reduces the expected number of transmissions over reliable multicast without FEC. This reduction does not come for free however, since there are processing requirements at the sender and the receivers for coding and decoding in the case of loss. We will now evaluate the processing load at sender and receivers and show how the use of integrated FEC affects the achievable end-host throughput of the reliable multicast connection. We will first present a reliable multicast protocol using integrated FEC, called NP, and then compare it with a generic version of a reliable multicast protocol without FEC, called N2.

5.1 Protocol NP

There are numerous ways to design a reliable multicast protocol with hybrid ARQ. The design choices are largely influenced by considerations for the specific type of application, e.g., file transfer or audio/video transport and its constraints such as high efficiency or low latency.

Protocol NP emphasizes efficiency at the expense of latency by only transmitting as many parities as are required to reconstruct a TG. NP could be used, for instance, by a reliable file transfer application. Protocol NP is similar to the Integrated FEC 2 scheme from Section 4.2, i.e., parity packets are retransmitted in response to received NAKs. A

single multicast group is used for the transmission of the data and the parity packets. The entire data block is broken up into multiple TGs, $TG_1, \ldots$, each consisting of k data packets.

Feedback from the receivers consists of the multicast transmission of NAKs coupled with NAK suppression as in SRM [5].

The transmission of $TG_i, i = 1, \ldots$, proceeds in rounds which are interleaved with the rounds of other TGs.

- Round 1: The k data packets of TG_i are sent
- Round $j > 1$: $l^{(j-1)}$ parities for TG_i are sent, where $l^{(j-1)}$ is the maximum number of packets over all receivers that still need to be received after $j-1$ rounds to reconstruct the k data packets of TG_i.

The Sender:

- Transmits the k data packets in transmission group TG_i.
- When done, the sender polls (`POLL(i,k)`) the receivers for feedback about the number of packets missing to reconstruct TG_i and continues by sending the data packets of TG_{i+1}.
- When the sender receives `NAK(i,l)` (see below), it interrupts sending data packets of TG_m if $m > i$. The sender then transmits l new parities for the data packets in TG_i and polls the receivers (`POLL(i,l)`) for feedback about the remaining number of packets required to reconstruct TG_i. It then resumes transmission of the interrupted transmission group TG_m.

The Receiver:

- Stores data packets and parities for TG_i until k packets are received, which allows the receiver to reconstruct TG_i.
- When a `POLL(i,s)` is received, the receiver computes the number of packets, l, needed to reconstruct transmission group TG_i and schedules a timeout for returning this information (`NAK(i,l)`) to occur in the interval $[(s-l)T_s, (s-l+1)T_s]$. Here the **slot size** T_s is chosen by taking the requirements of the application (low latency, high efficiency) into account.
- When the timeout for `NAK(i,l)` occurs, `NAK(i,l)` is re-sent. The timer for `NAK(i,l)` is canceled on the reception of `NAK(i,m)` with $m \geq l$.

With our slotting and damping mechanism the sender will ideally receive a single `NAK(i,l)` after every round as a reply that indicates the *maximum number* of packets needed by any receiver to reconstruct TG_i.

Protocol NP is similar to protocol N2 of [20] in several aspects: feedback is receiver-initiated and NAKs are sent via multicast. A receiver receiving a NAK for a particular round will not generate a NAK for that round. The major differences between NP and N2 are that NP requires feedback for a TG of k packets rather than for *individual* packets and that NP transmits parity packets for loss recovery while N2 retransmits the original packets that are lost.

In order to quantify the performance impact of the differences between N2 and NP, we compare their processing rates at the sender and receiver and their throughput for the case of a one-to-many transmission. Let Λ_s^w, Λ_r^w be the per-packet send and receive processing rates of protocol $w \in \{N2, NP\}$. The achievable end-system throughput Λ_o^w is defined as the minimum of the sender and receiver processing rates:

$$\Lambda_o^w = min\{\Lambda_s^w, \Lambda_r^w\} \tag{10}$$

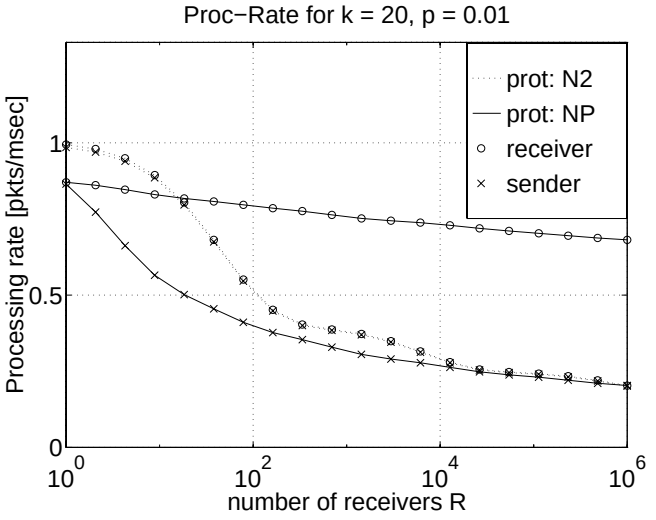

Figure 17: Processing rates at sender and receiver for protocols N2 and NP with packet size $P = 2$ KByte, $k = 20$, and $p = 0.01$.

In the following, we compare the processing rates for protocols N2 and NP as a function of the number of receivers. The processing rates $\Lambda_s^{N2}, \Lambda_r^{N2}$ were computed in [20]. The computation of the processing rates $\Lambda_s^{NP}, \Lambda_r^{NP}$ is given in the appendix. To obtain the results, we used the same values for the various processing times as [20] along with our own measured values (on the same DECstation 5000/200 running ULTRIX 4.2a as in [20]) for the encoding and decoding times based on the coder reported in [14]. In our throughput calculations we use $E[X_p] = E[Y_p] = 1000\mu$sec (average processing times for sending or receiving a 2 KByte packet) and $E[X_n] = E[Y_n] = E[Y_n'] = 500\mu$sec (average processing time to send or receive a NAK). We use $E[X_t] = E[Y_t] = 24\mu$sec for the timer overhead. We measured as coding constants $c_e = 700\mu$sec and the decoding constant as $c_d = 720\mu$sec for 2 KByte packets and a symbol size of $m = 8$. The reader is referred to the appendix for definitions of the above quantities.

In Figure 17, we see that the sender and receiver processing rates are nearly identical for protocol N2. The processing rates are largely determined by the mean number of transmissions and NAKs to be processed per packet (see Eq. (11) and (12)). They decrease as the number of receivers increases due to the fact that the mean number of transmissions per packet increases.

For NP, the processing rate at the sender is largely determined by the packet processing times and the encoding

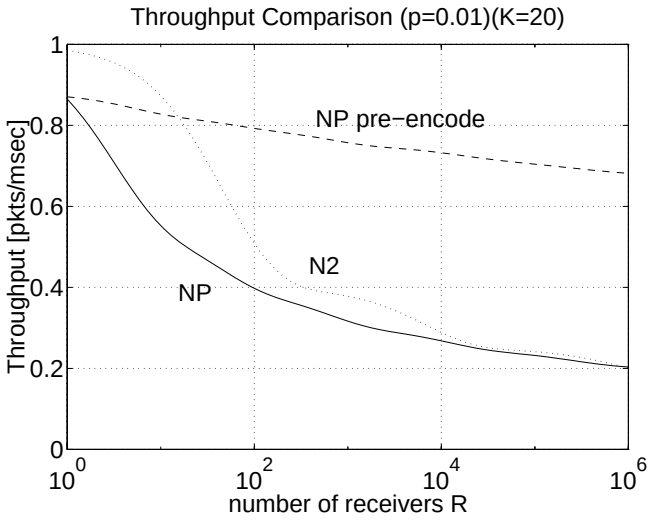

Figure 18: Throughput [pkts/msec] for N2 and for NP with and without pre-encoding for packet size $P = 2$ KByte, $k = 20$, and $p = 0.01$.

times, both of which *depend linearly* on the mean number of transmissions (see Eq. (14) and (16)). At the receiver, the processing rate is largely determined by the decoding time, which is *independent* of the number of receivers (Eq. (17)) and the packet processing time, which increases linearly with the mean number of transmissions.

The processing overhead due to FEC is much higher at the sender than at the receivers: the sender must encode a number of parities, (with expected value $E[M^{NP}] - 1$) sufficient to allow the reconstruction of the data packets of a TG by *all* receivers. An individual receiver needs to decode (reconstruct) an average of $k \cdot p$ packets per TG. Therefore, the receiver processing rate is much smaller than the sender processing rate.

Protocol NP contains two improvements over N2: loss recovery via parity retransmission and feedback reduction due to the use of a single NAK *per transmission round* instead of per missing packet. By slightly modifying the analysis in the appendix, (Eq. (14) and (15)) we can obtain the processing rates for the case that *one NAK is returned per missing packet*. The results indicate that reducing the NAKs to one per transmission round, as does protocol NP, has only a minor effect on the processing rates; the sender processing rate does not change and the receiver processing rate decreases only slightly for a very large number of receivers.

The processing rates obtained show that NP scales with the number of receivers, R, since the processing bottleneck is a single point – the sender. For N2, this is not the case, since sender and receivers have the same processing rates. If required, there are some simple solutions to match the speed of the NP sender and the NP receivers: *(i)* the sender can **pre-encode** the packets off-line and store the parity data together with the original data prior to transmission on disk, *(ii)* a more powerful machine can be used at the sender, or *(iii)* dedicated hardware can be used for encoding. Figure 18 compares the throughput given by Equation (10) for N2 and NP with and without pre-encoding, for $k = 20$, $p = 0.01$. It demonstrates the extent to which encoding impacts the performance of the NP protocol. It can be seen that the throughput of NP with pre-encoding is higher than the throughput of N2 and NP without pre-encoding, even for a small number of receivers.

6 Summary

Using FEC in an integrated fashion provides five major benefits:

- Integrated FEC shifts resource usage from the network to the end-systems: the number of transmissions is reduced and, therefore, the network bandwidth used as well – at the extra cost of coding/decoding in the end-systems.
- The achievable end-system throughput for protocols based on integrated FEC is higher than for non-FEC protocols when data is pre-encoded.
- Error-control feedback is reduced: feedback is returned for each transmission group (consisting of k packets) rather than for each packet.
- A moderate TG size of $k = 20$ will tolerate burst losses, even without interleaving.
- Scalability with the number of receivers is achieved for reliable multicast up to 1 million receivers.

We can draw the following conclusions:

- Integrated FEC dramatically reduces the mean number of transmissions as compared to the use of no FEC.
- Integrated FEC is better than layered FEC for all parameters. In addition, low redundancy is sufficient to achieve idealized integrated FEC.
- Layered FEC can reduce the number of transmissions in the case of large receiver populations. However, unlike integrated FEC, its performance is sensitive to the coding parameters and the presence/absence of burst losses. Layered FEC may be useful for applications with delay constraints; this is a topic for future work.
- High loss receivers determine the performance of a reliable multicast, even if the fraction of high loss receivers among all receivers is very small.
- The repair efficiency of FEC is in the case of shared loss not as high as in the case of independent loss. For a given number of receivers, shared loss can be modeled by a reduced number of receivers that suffer independent losses.
- For burst loss, layered FEC can be worse than no FEC. When losses are bursty, the performance of integrated FEC decreases, especially for small values of k. In this case, interleaving can improve performance. However, integrated FEC with a large TG size k does not need interleaving.
- For protocols based on integrated FEC (such as NP) the sender is the bottleneck. Pre-encoding the parity packets or using a higher performance sender machine

can yield an end-system throughput that is three times higher than for a reliable multicast protocol with no FEC, even when receivers decode online.

Acknowledgment

We are indebted to Brian DeCleene for suggesting the study of the layered and integrated approach. The authors are grateful to Stephan Röösli for measuring the performance of the FEC coder and decoder and to Jean Bolot for his suggestions on the burst loss model. Our thanks to Dan Rubenstein and the anonymous reviewers for many helpful comments.

References

[1] S. Lin and D. J. Costello, *Error Correcting Coding: Fundamentals and Applications*, Prentice Hall, Englewood Cliffs, NJ, 1983.

[2] S. Lin, D. J. Costello, and M. J. Miller, "Automatic-repeat-request error-control schemes.", *IEEE Commun. Magazine*, 22(12):5–17, 1984.

[3] J. Metzner, "An Improved Broadcast Retransmission Protocol", *IEEE Transactions on Communications*, COM-32(6):679–683, June 1984.

[4] C. Huitema, "The case for packet level FEC", *Proceedings of IFIP 5th International Workshop on Protocols for High Speed Networks (PfHSN'96)*, pp. 110–120, INRIA, Sophia Antipolis, FRANCE, October 1996, Chapman & Hall.

[5] S. Floyd, V. Jacobsen, S. McCanne, C.-G. Liu, and L. Zhang, "A Reliable Multicast Framework for Lightweight Sessions and Application Level Framing", *Proceedings of SIGCOMM'95*, volume 25, pp. 342–356, ACM, October 1995.

[6] T. W. Strayer, B. J. Dempsey, and A. C. Weaver, *XTP - THE XPRESS TRANSFER PROTOCOL*, Addison-Wesley, 1992.

[7] S. Paul, K. K. Sabnani, J. C. Lin, and S. Bhattacharyya, "Reliable Multicast Transport Protocol (RMTP)", *IEEE Journal on Selected Areas in Communications, special issue on Network Support for Multipoint Communication*, 15(3):407 – 421, April 1997.

[8] M. Hofmann, "A Generic Concept for Large-Scale Multicast", B. Plattner, Ed., *Proc. International Zuerich Seminar*, volume 1044 of *LNCS*, pp. 95–106, Springer Verlag, February 1996.

[9] R. Yavatkar, J. Griffoen, and M. Sudan, "A reliable Dissemination Protocol for Interactive Collaborative Applications", *Proceedings of ACM Multimedia*, pp. 333–344, San Francisco, CA USA, 1995, ACM.

[10] K. Sakakibara and M. Kasahara, "A Multicast Hybrid ARQ Scheme using MDS Codes and GMD Decoding", *IEEE Transactions on Communications*, 43(12):2933–2939, December 1995.

[11] R. H. Deng, "Hybrid ARQ Schemes for Point-to-Multipoint Communication over Nonstationary Broadcast Channels", *IEEE Transactions on Communications*, COM-41(9):1379–1387, September 1993.

[12] J. Nonnenmacher and E. W. Biersack, "Reliable Multicast: Where to use FEC", *Proceedings of IFIP 5th International Workshop on Protocols for High Speed Networks (PfHSN'96)*, pp. 134–148, INRIA, Sophia Antipolis, FRANCE, October 1996, Chapman & Hall.

[13] A. J. McAuley, "Reliable Broadband Communications Using a Burst Erasure Correcting Code", *Proc. ACM SIGCOMM 90*, pp. 287–306, Philadelphia, PA, September 1990.

[14] L. Rizzo, "Effective erasure codes for reliable computer communication protocols", *Computer Communication Review*, 27(2):24–36, April 1997.

[15] A. Jean-Marie, "On the Average Number of Retransmissions in Multicast Applications", available from the author, INRIA, Sophia Antipolis, France, Aug. 1997.

[16] J. Nonnenmacher and E. W. Biersack, "The Impact of Routing on Multicast Error Recovery", *To appear in: Computer Communications*, 1998.

[17] P. Bhagwat, P. P. Mishra, and S. K. Tripathi, "Effect of Topology on Performance of Reliable Multicast Communication", *Proceedings of IEEE INFOCOM'94*, volume 2, pp. 602–609, Toronto, Ontario, Canada, June 1994.

[18] P. Morse, *Queues, Inventories, and Maintenance*, John Wiley, 1958.

[19] J. C. Bolot, "Analysis and control of audio packet loss in the Internet", T. D. C. Little and R. Gusella, Eds., *5th Workshop on Network and Operating System Support for Digital Audio and Video*, volume 1018 of *LNCS*, pp. 163–174, Springer Verlag, Heidelberg, Germany, April 1995.

[20] D. Towsley, J. Kurose, and S. Pingali, "A Comparison of Sender-Initiated and Receiver-Initiated Reliable Multicast Protocols", *IEEE Journal on Selected Areas in Communications*, 15(3):398–406, 1997.

[21] E. Ayanoglu, R. D. Gitlin, and N. C. Oguz, "Performance Improvement in Broadband Networks using Forward Error Correction for Lost Packet Recovery", *Journal of High Speed Networks*, 2:287–303, 1993.

[22] D. Rubenstein, "Using Packet-level FEC with Real-time Data Delivery", available from the author, CS department, University of Massachusetts, Amherst, May 1997.

Appendix

Let us first recall from [20] the equations for the processing rates for protocol N2.

$$\begin{aligned} 1/\Lambda_s^{N2} &= E[X^{N2}] \\ &= E[M^{N2}]E[X_p] + (E[M^{N2}] - 1)E[X_n] \end{aligned} \tag{11}$$

$$
\begin{aligned}
1/\Lambda_r^{N2} &= E[Y^{N2}] \qquad (12)\\
&= E[M^{N2}](1-p)E[Y_p]\\
&\quad +(E[M^{N2}]-1)\left(\frac{1}{R}E[Y_n]+\frac{R-1}{R}E[Y_n^{'}]\right)\\
&\quad +E[(M_r-2)^+]E[Y_t]
\end{aligned}
$$

We can derive the processing rates for NP in a similar way to N2, taking into account the time for encoding and decoding and the fact that feedback and retransmissions are performed for entire TGs. We define the following variables:

X_e	time to encode a packet at the sender, which is a function of k and h
c_e	constant encoding time factor
X_p	time to process the transmission of a packet
X_n	time to process a NAK at the sender
Y_p, Y_t	time to process a packet or timeout at a receiver
Y_d	time to decode a packet at a receiver, which is a function of k and l
c_d	constant decoding time factor
Y_n	time to process and transmit a NAK at a receiver
$Y_n^{'}$	time to receive and process a NAK at a receiver
l	number of lost packets in a transmission group
M_r	number of transmissions necessary for receiver r to successfully receive a packet
M^w	number of transmissions for all receivers to successfully receive a packet, $w \in \{N2, NP\}$
T_r	number of transmission rounds necessary for receiver r to successfully receive a TG
T	number of transmission rounds for all receivers to successfully receive a TG
X^w, Y^w	send and receive per packet processing times of protocol $w \in \{N2, NP\}$
Λ_s^w, Λ_s^w	send and receive per packet processing rates of protocol $w \in \{N2, NP\}$
Λ_o^w	throughput of protocol $w \in \{N2, NP\}$ for a one-to-many transmission

The parameters p and k remain as before. To simplify the analysis we make the following assumptions:

- Losses among receivers are independent.
- The sender never runs out of parities. Otherwise, receivers requiring more than h parities can be ejected.
- Per transmission round always only one NAK is sent. NAKs are never lost.
- The buffer at the receivers is large enough to store the packets from all the TGs that cannot yet be reconstructed.

$$\Lambda_o^{NP} = min\{\Lambda_s^{NP}, \Lambda_r^{NP}\} \qquad (13)$$

With

$$
\begin{aligned}
1/\Lambda_s^{NP} &= E[X^{NP}] \qquad (14)\\
&= E[X_e] + E[M^{NP}]E[X_p] + \frac{E[T]-1}{k}E[X_n]
\end{aligned}
$$

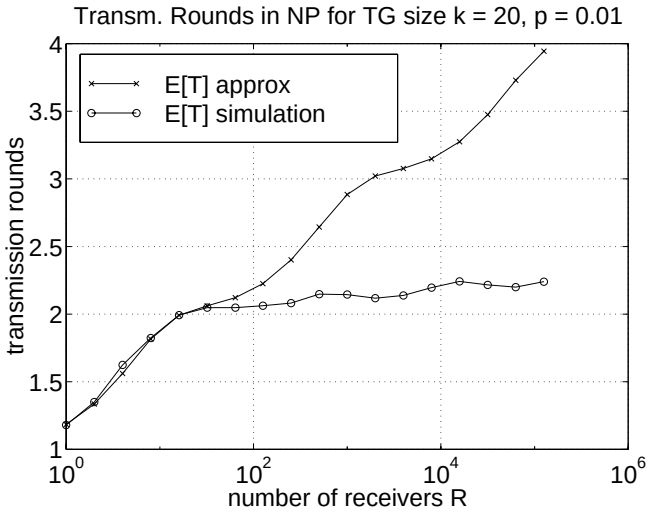

Figure 19: The number of transmission rounds: Simulation and the analytical approximation for protocol NP with parameters $k = 20$, $p = 0.01$ for independent loss.

$$
\begin{aligned}
1/\Lambda_r^{NP} &= E[Y^{NP}] \qquad (15)\\
&= E[M^{NP}](1-p)E[Y_p]\\
&\quad +((E[T]-1)/k)\left(\frac{1}{R}E[Y_n]+\frac{R-1}{R}E[Y_n^{'}]\right)\\
&\quad +E[(T_r-2)^+]E[Y_t]+E[Y_d]
\end{aligned}
$$

$E[M^{NP}]$ is computed as $E[M]$ of Eq. (7).

For the RSE coder presented in [14] per packet coding and decoding times are:

$$
\begin{aligned}
E[X_e] &= k\cdot(E[M^{NP}]-1)\cdot c_e \qquad (16)\\
E[Y_d] &= k\cdot p\cdot c_d \qquad (17)
\end{aligned}
$$

where $k \cdot p$ denotes the mean number of data packets per transmission group that are lost and need to be reconstructed and c_e and c_d are constants for encoding and decoding that depend on the particular hardware, the symbol size m and the the size P of the packets.

The mean number of transmission rounds is

$$
\begin{aligned}
E[T_r] &= \sum_{m=0}^{\infty}(1-P[T_r \le m])\\
E[(T_r-2)^+] &= E[T_r]-P[T_r=1]-2P[T_r>1]\\
E[T] &= \sum_{m=0}^{\infty}(1-P[T \le m]) \qquad (18)
\end{aligned}
$$

with $P[T \le m] = P[T_r \le m]^R,\ m = 1, 2, 3, ...$

For $P[T_r \le m]$ we use

$$P[T_r \le m] \approx (1-p^m)^k,\ m = 1, 2, 3, ..$$

from the expressions derived in [21], which assumes that the number of parity packets sent during a transmission round is the same as the number of parities needed by receiver r. Since the sender will, however, send the maximum

number of parities required by any receiver, this assumption will yield an upper bound on the expected number of transmission rounds.

Figure 19 shows that the analytical approximation provides is close to the exact value of $E[T]$ for $R < 50$. For large numbers of receivers the approximation gives a loose upper bound of the exact value of $E[T]$. The derivation of the exact number of transmission rounds can be found in [22], however its use is computationally very intensive.

Jörg Nonnenmacher (S) received his M.Sc. degree in Computer Science from the University of Karlsruhe, Germany, in 1995 and the Ph.D. degree in 1998 from EPFL, Lausanne, Switzerland. He conducts research at Institut Eurecom, Sophia Antipolis, France, where he joined 1995 as a research scientist and teaching assistant.

His current research interests include control in large distributed systems, and new applications for integrated networks.

He is a student member of the IEEE.

Ernst Biersack (M) received his M.S. and Ph.D. degrees in Computer Science from the Technische Universität München, Munich, Germany.

From 1983 to 1989 he was a research scientist at the Technische Universität München. From March 1989 to February 1992 he was a Member of Technical Staff with the Computer Communications Research District of Bell Communications Research.

Since March 1992 he has been an Associate Professor in Telecommunications at Institut Eurecom, in Sophia Antipolis, France. His current research is on "Architectures for a Scalable High-Performance Video Servers" and "Reliable Multicast Transmission". For his work on synchronization in video servers he received the Outstanding Paper Award of the IEEE Conference on Multimedia Computing & Systems 1996.

He currently serves on the editorial board of IEEE Network Magazine, ACM Multimedia Systems, and ACM/IEEE Transactions on Networking. Mr. Biersack is a member of the IEEE and ACM.

Don Towsley (F) received the B.A. degree in physics and the Ph.D. degree in Computer Science from University of Texas in 1971 and 1975 respectively. From 1976 to 1985 he was a member of the faculty of the Department of Electrical and Computer Engineering at the University of Massachusetts, Amherst. He is currently a Professor of Computer Science at the University of Massachusetts. During 1982-1983, he was a Visiting Scientist at the IBM T.J. Watson Research Center, Yorktown Heights, NY. and during the year 1989-1990, he was a Visiting Professor at the Laboratoire MASI, Paris, France. His research interests include high speed networks and multimedia systems.

He currently serves on the Editorial board of *Performance Evaluation* and has previously served on several editorial boards including those of the *IEEE Transactions on Communications* and *IEEE/ACM Transactions on Networking*. He was a Program Co-chair of the joint ACM SIGMETRICS and PERFORMANCE '92 conference. He is a member of ACM, ORSA and the IFIP Working Groups 6.3 and 7.3.

He has received two best conference paper awards from ACM SIGMETRICS and has been elected Fellow of both the ACM and IEEE.