Parity-Based Loss Recovery for Reliable Multicast Transmission

Jörg Nonnenmacher, Ernst Biersack,
Institut Eurecom
06904 Sophia-Antipolis Cedex France
{nonnen,erbi}@eurecom.fr

Don Towsley*
Dept. of Computer Science
Univ. of Massachusetts,
Amherst, MA 01003-4610, USA
towsley@cs.umass.edu

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 types of loss behaviors (spatially or temporally correlated loss, homogeneous or heterogeneous loss) and loss rates for up to 10^6 receivers. Our results show that introducing FEC as a layer below ARQ can improve multicast transmission efficiency and scalability and that there are substantial additional improvements when the two 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 deal with 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. If the amount of original data lost is not more than the amount of parity data sent, the parity data can be used to reconstruct the lost original data.

*The work of D. Towsley was supported in part by INRIA and 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.

FEC by itself cannot provide full reliability. However, when coupled with ARQ, FEC can 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** [1].

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 literature on the subject of reliable multicast - several of which focus on the use of FEC. An early paper by Metzner [2] 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 the 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 [3] studied the performance 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 [4, 5] or introduce a hierarchy [6, 7, 8]. Both solutions 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 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 [2, 9, 10, 3, 11]. 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 [12], 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 data packets $d_1, \ldots, d_k$ as elements of the Galois field $GF(2^P)$ [13] and define the polynomial $F(X)$ as

$$F(X) = d_1 + d_2X^1 + \ldots + d_kX^{k-1}. \tag{1}$$

If α is the primitive element of $GF(2^P)$, the RSE encoder computes $p_j = F(\alpha^{j-1})$ for $j \in \{1, \ldots, n-k\}$.

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 an **FEC block**. Sending the original data as the first k packets of the FEC block simplifies decoding:

- If all the 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 the parity packets for loss recovery instead of retransmitting the lost 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.*
- Improved scalability in terms of group size:
  In an ARQ scheme retransmitting the original packets that are lost the sender needs to know the sequence numbers of the lost packets. Using parity packets for loss repair, the sender needs only to know the maximum number of packets lost by any receiver but not their sequence numbers. Feedback is reduced from per–packet feedback to per–TG feedback.
- Reduction of unnecessary receptions:
  Multicasting retransmissions for loss recovery results in unnecessary receptions at all receivers that do not need the retransmission. Such duplicate packets waste transmission bandwidth and processing capacity. As we will show later, the number of duplicate packets received due to retransmissions by any receiver can be reduced nearly to zero with parity transmission.

2.2 Practical Aspects

The size P of a packet will typically be in 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 [12] uses a symbol size m with $m = 8$ or $m = 32$. The software (SW) implementation of the RSE coder by Rizzo [14] uses $m = 8$.

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 packet (see Figure 2 of [12]). For example, we perform RSE on the first m-bit symbol of each of the k packets and obtain $n-k$ m-bit parity packets. We repeat the same on the 2nd m bit symbol of the k packets and so on.

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

Figure 1 shows the coding and decoding throughput of the RSE coder by Rizzo. We measured the throughput on a Pentium PC 133 running Linux with packet size $P = 1$ KByte. The encoding throughput denotes the number of data packets that can be processed per second, given that $h = n - k$ parity packets are produced for every k original data packets. The decoding throughput denotes the rate at which data packets can be reconstructed per second, given that h out of every k data packets are lost and must be reconstructed. We see that the throughput is inversely proportional to $h \cdot k$.

The throughput of the coder is largely sufficient for many current applications. It can be seen that $h = 1$ parity packet for $k = 7$ original packets (14.3% redundancy) is coded with a speed of 8000 original packets per second, which means that the first redundant packet for a transmission group of size $k = 7$ is available after $0.125ms$. Such high performance and the fact that the bandwidth requirements of many current multicast applications are typically less than 100 KByte/s, suggests that coding will not affect the packet sending rate and, hence, 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

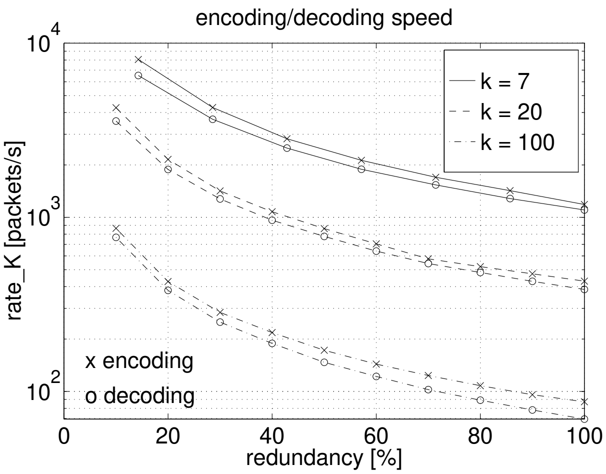

Figure 1: Coding and decoding rates in packet/s with respect to the redundancy h/k and the transmission group size k.

(RM). The second approach is to integrate FEC with reliable multicast and place it into one 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 reliable multicast. 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 can benefit from a more reliable network. Huitema [3] has established the benefits of layered FEC in terms of reducing the network traffic. We will review this work in Section 3.1.

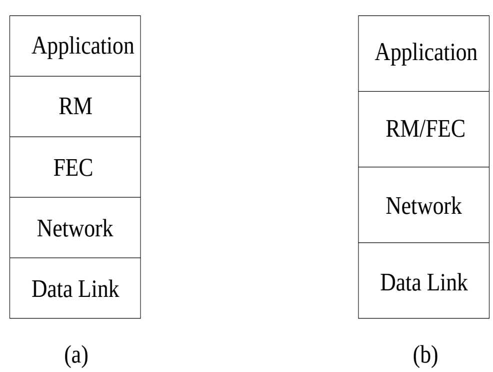

Figure 2: (a) Layered FEC. (b) Integrated FEC.

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 under layered FEC, integrated FEC, and no FEC. This metric is important because it reflects directly the network bandwidth required to support reliable 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 a reduction in the required number of transmissions will often lead to a reduction in latency.

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 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 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 group.

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, packet i from a TG is lost at the RM receiver if it is lost by the FEC receiver and more than $h-1 = n-k-1$ out of the other $n-1$ packets from the FEC block are lost. This was first proposed and studied in [3]. Therefore the packet loss probability at the RM receiver is for $1 \leq k \leq n$ given by

$$q(k,n,p) = p\left(1 - \sum_{j=0}^{n-k-1}\binom{n-1}{j}p^j(1-p)^{n-j-1}\right) \tag{2}$$

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 $1-q(k,n,p)^i$. The cumulative distribution of M' is therefore:

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

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

$$\begin{aligned} E[M] &= n/k \cdot E[M'] \\ &= n/k \cdot \sum_{i=0}^{\infty} 1 - P(M' \leq i) \end{aligned} \tag{3}$$

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 is 2 and 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 [3]. Second, we observe that the number of parity packets needs to be matched to the TG size. For example, a TG of size

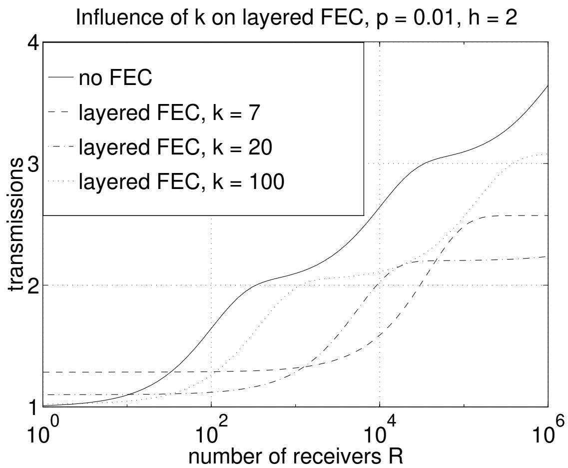

Figure 3: Non FEC versus layered FEC with $h = 2$ parity packets for different TG sizes $k = 7, 20, 100$ and loss probability $p = 10^{-2}$.

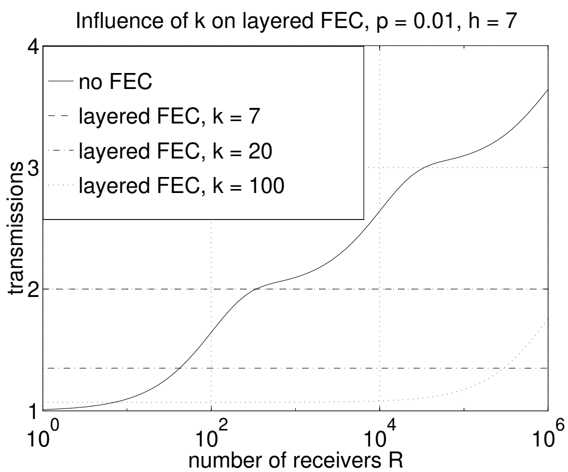

Figure 4: Non FEC versus layered FEC with $h = 7$ parity packets for different TG sizes $k = 7, 20, 100$ and loss probability $p = 10^{-2}$.

$k = 100$ with $h = 2$ parity packets performs worse than TGs of sizes 7 or 20 with the same number of parity packets. On the other hand a TG of size 100 coupled with 7 parity packets performs better than TGs of sizes 7 and 20 for receiver populations in the 1 - 200,000 range.

We have observed similar behavior for a wide range of packet loss probabilities.

3.2 Integrated FEC

We 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.
- Each time that a receiver detects a missing packet, it requests a new parity packet from the sender until it has 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

$$\begin{aligned} P(L_r = 0) &= \sum_{j=0}^{a} \binom{k+a}{j} p^j (1-p)^{k+a-j}, \\ P(L_r = m) &= \binom{k+a+m-1}{k-1} p^{m+a} (1-p)^k, \\ & \qquad m = 1, \ldots \end{aligned}$$

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

$$P(L \leq m) = [P(L_r \leq m)]^R, \quad m = 0, 1, \ldots \tag{4}$$

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

$$E[L] = \sum_{m=0}^{\infty} (1 - P(L \leq m)). \tag{5}$$

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

$$E[M] = (E[L] + k + a)/k. \tag{6}$$

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 require the transmission of n packets each. 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

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

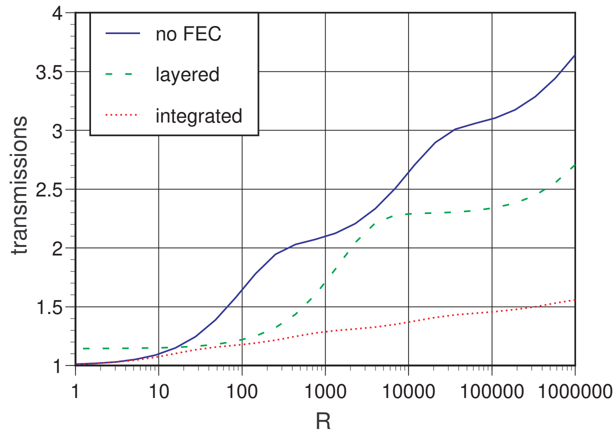

Figure 5: $E[M]$ versus R for TG size of 7 and loss probability of $p = 10^{-2}$. Layered FEC versus integrated FEC.

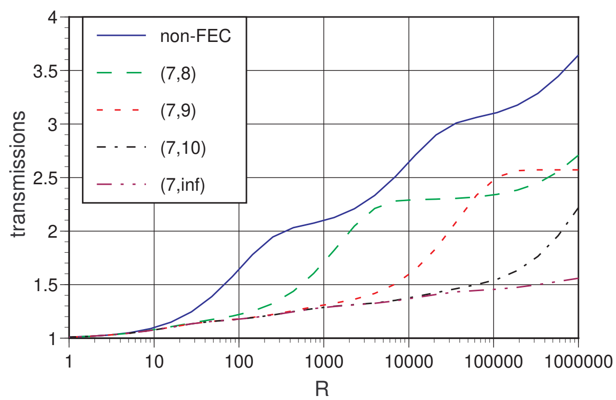

Figure 6: $E[M]$ versus R for TG size of 7 and loss probability of $p = 10^{-2}$. Integrated FEC with $k = 7$ for different values of h.

where $q(k,n,p)$ is computed using expression (2) and the properly conditioned version of $P(L \leq m)$ given in (4) is $P(L \leq m \,|\, L \leq n)$.

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 7, 3 parity packets suffice to attain the lower bound for receiver populations of up to 100,000 to 200,000. Henceforth, we will use the lower bound when comparing integrated FEC to other approaches.

Last, Figures 7 and 8 illustrate the influence of the TG size, k, on the performance of integrated FEC. They show that increasing the transmission group size reduces the number of transmissions for integrated FEC nearly down to one, even for a large number of receivers. On the other hand a relatively small performance gain is achieved by increasing the number of data packets, k, in a block.

We observe from Figure 8 that this behavior is insensitive to the loss probability – a large increase in the transmission group size has little effect on the performance of integrated FEC.

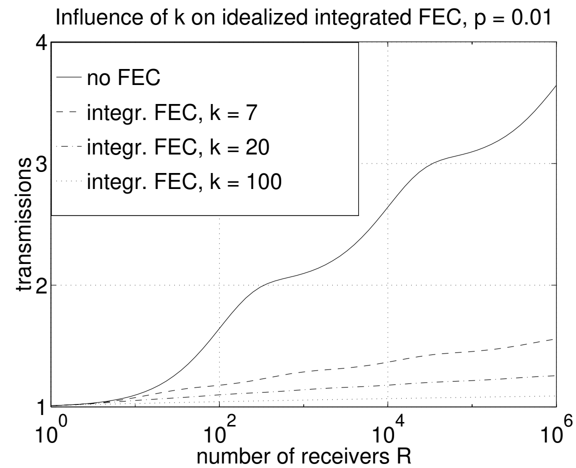

Figure 7: Influence of number of receivers on integrated FEC for different TG sizes for packet loss probability $p = 10^{-2}$.

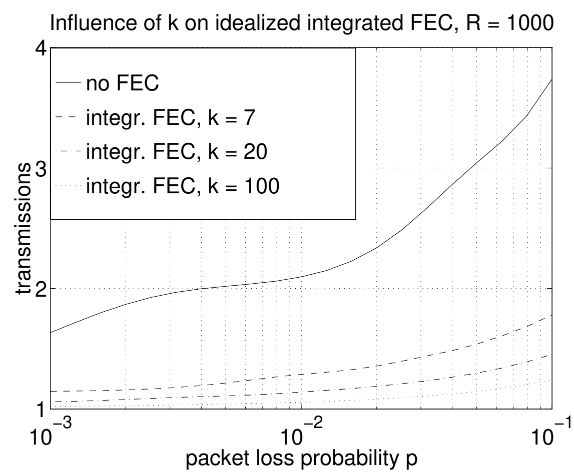

Figure 8: Influence of loss probability on integrated FEC for different TG sizes for $R = 1000$ receivers.

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 we have for layered FEC

$$E[M] = \sum_{i=0}^{\infty}(1 - \prod_{r=1}^{R}(1 - q(k,n,p(r))^i))n/k \qquad (7)$$

In the case of integrated FEC, $E[M]$ is given by equation (6) with

$$P(L \leq m) = \prod_{r=1}^{R} P(L_r \leq m) \qquad (8)$$

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 percentage 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 $1\%, 5\%$ 25% of the whole group.

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 1 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 100 has much less effect on the expected number of transmissions. Last, the presence of high loss receivers has a greater effect in the case of integrated FEC than no FEC.

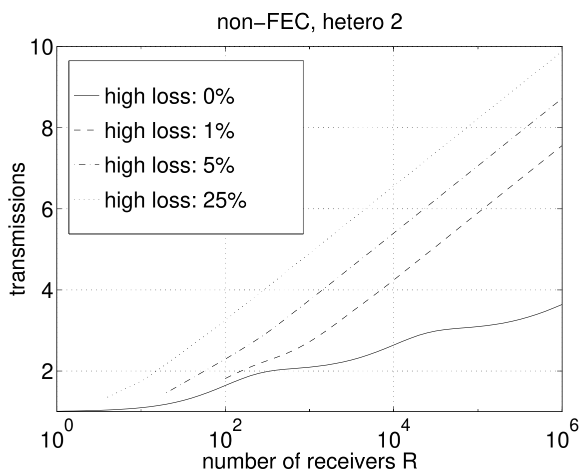

Figure 9: Reliable multicast for different heterogenities without FEC.

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 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 focusses on spatial loss correlation and Section 4.2 focusses on temporal loss correlation.

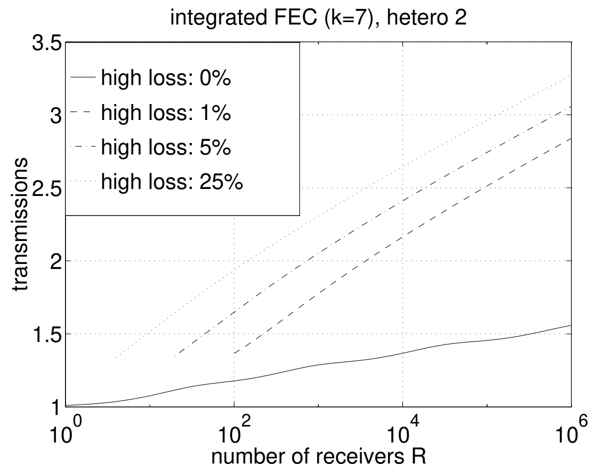

Figure 10: Reliable multicast for different heterogenities with integrated FEC.

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 one or more 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 order to investigate the impact of shared loss we will consider the following two loss models:

- **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.
- **FBT shared loss**: We consider a full binary tree (FBT) of height d, where losses occur as independent events at each node (including source and leaves) with the probability p_{node}. The source is the root of the tree and the receivers are the leaves. Here p_{node} is chosen such that the loss probability at each receiver is p:

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

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 [15]. Because the calculation of this quantity is computationally intensive even for $R = 64$ receivers, we use simulation to investigate the impact of shared loss for large numbers of receivers: $R = 2^d$, $d = 0, \ldots, 17$. The packet loss probability is $p = 10^{-2}$, FEC was coded for transmission groups of size $k = 7$ and one parity packet is transmitted ($h = 1$) in the case of layered FEC.

Layered FEC transmits parities whether or not a loss occurs. A transmitted parity is able to repair different losses at different receivers. When shared loss occurs, this parity does not exhibit the same repair efficiency 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 will be amortized by the repair efficiency for a number of receivers $R > 60$ in the case of shared loss, while in the case of independent loss layered FEC is already efficient for $R > 20$. Hence, for real trees (here modeled by a full binary tree) layered FEC is worthwhile only for multicast group sizes that are larger than those that the results for independent loss suggest.

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 that the 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, as mentioned previously, receiver group sizes need to be larger before the benefits of layered FEC appear. Furthermore, the benefits of integrated FEC, while remaining substantial, are not as great when losses are shared.

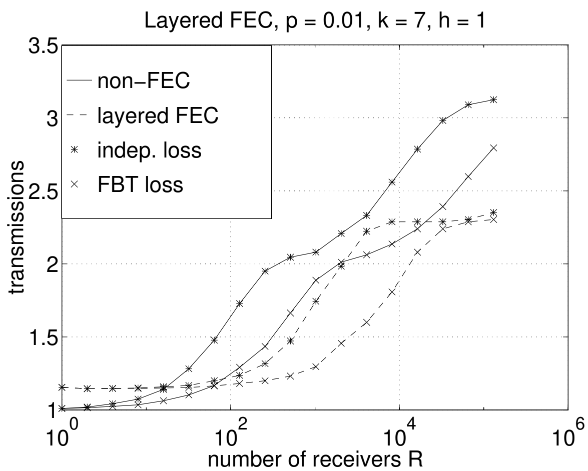

Figure 11: Layered FEC compared with non-FEC for Independent and for FBT Shared Loss.

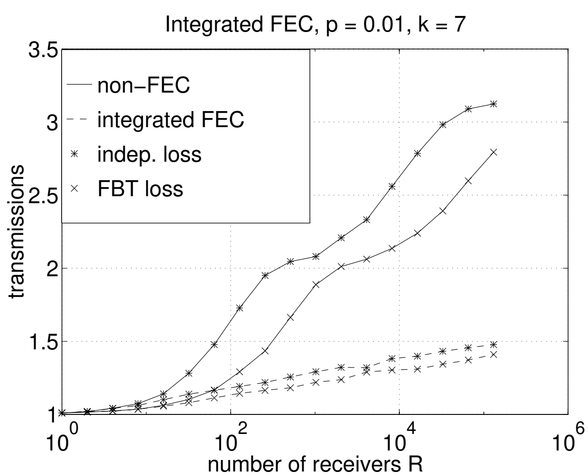

Figure 12: Integrated FEC compared with non-FEC for Independent and for FBT Shared Loss.

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 in either the presence or absence of FEC 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.

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, $p_{i,j}(t) = P(X_{\tau+t} = j \,|\, X_\tau = i)$. Expressions for these probabilities can be found in [16, ch. 6].

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.

When burst losses occur, the timing of the retransmissions has an influence on the performance of the 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. When a receiver has received enough parity packets, it leaves the multicast group. In this scheme 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.
- **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.

A well-known technique that allows FEC to deal with burst loss is interleaving. Under interleaving the sender spreads the transmission of a FEC block over an interval that is longer than the loss burst length. 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 transmission groups can be sent simultaneously in an interleaved manner.

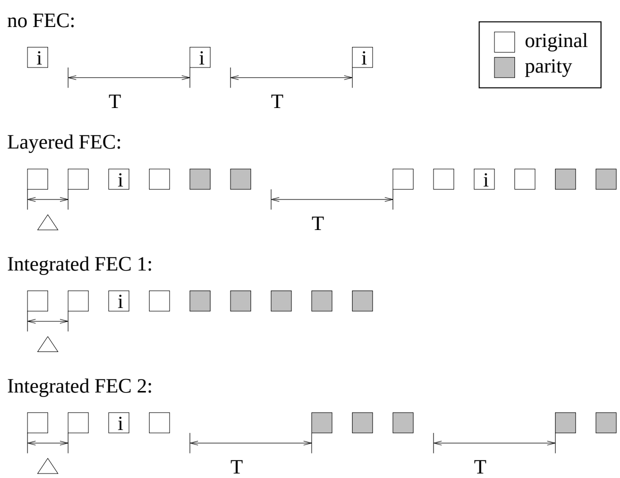

Figure 13: Timing considerations of the different approaches.

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:

$$\mu_0 = -\pi_1 \lambda \log(1 - \frac{1}{\bar{b}}) \qquad \mu_1 = \mu_0 \frac{1-p}{p}$$

We use simulation to show 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 [17] for a loaded IP path between Sophia Antipolis (INRIA) and London (UCL). 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 for independent and for burst loss among

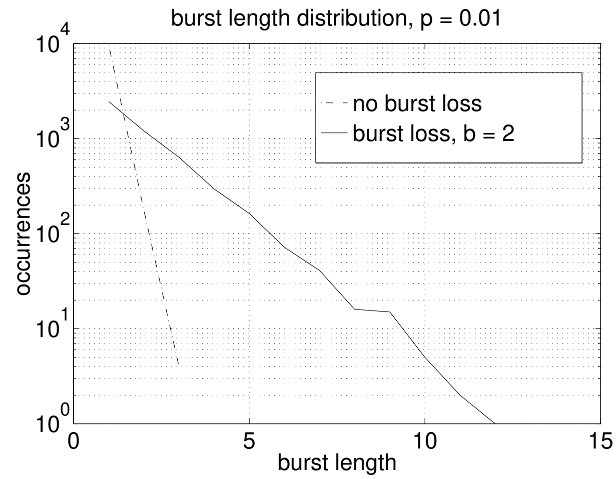

Figure 14: Distribution of number of consecutive losses at one receiver, for no burst and burst ($\bar{b} = 2$) for a packet loss probability of $p = 0.01$.

packets 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. It is possible that a larger TG coupled with more parity packets will permit layered FEC to outperform no FEC. However increasing the transmission group size is not desirable, since this will lead to encoding/decoding latencies that may no longer be transparent to the RM layer.

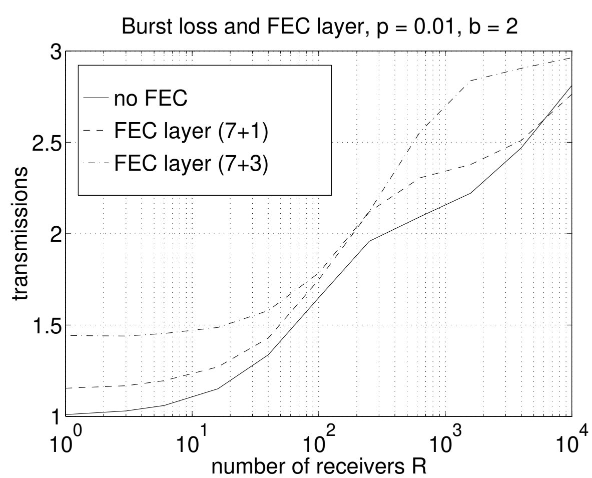

Figure 15: Comparison of reliable multicast without FEC and with layered FEC for low ($h = 1$) and high ($h = 3$) redundancy for $k = 7$, $p = 0.01$ and $\bar{b} = 2$.

Although large TGs 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 transmission group sizes ($k = 7, 20, 100$). For a small transmission group 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

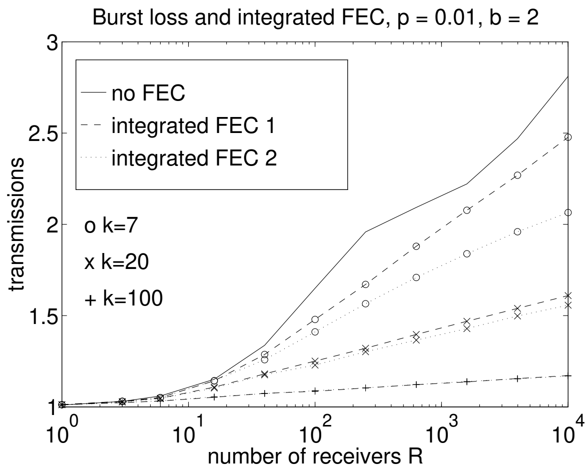

Figure 16: Comparison of integrated FEC 1 and integrated FEC 2 for transmission group sizes $k = 7, 20, 100$, $p = 0.01$ and $\bar{b} = 2$.

better than integrated FEC 1 for $k = 7$, since parity packets belonging to the same transmission group are spread out over time (see Figure 13) and are more likely to bridge a loss period. Figure 16 also shows that increasing the transmission group 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 transmission group size is sufficient to be resistant against 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 reliable multicast with FEC 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.

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.

The protocol NP emphases efficiency at the expense of latency by only transmitting as many parities as there are parities needed 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 transmission groups consisting of k data packets and $n-k$ parities.

Feedback from the receivers consists of the multicast transmission of NAKs coupled with NAK suppression as in SRM [4].

The transmission of TG_i proceeds in rounds, which are interleaved with the transmission of packets from other transmission groups.

- 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 lost in the previous round $j-1$ for 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 $TG_m, m > i$. The sender then transmits l parities for the data packets in TG_i and polls the receivers (`POLL(i,l)`) for feedback about the 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 l of packets needed to reconstruct transmission group TG_i and schedules a timeout for returning this information (`NAK(i,l)`) to occur in the interval $[(s-l)T_s, (s-l+1)T_s]$. The **slot size** T_s needs to be chosen appropriately 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[1] 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 in several aspects similar to the protocol N2 of [18]: feedback is receiver-initiated, NAKs are sent via multicast. A receiver receiving a NAK will not generate a NAK for the same round. The major differences between NP and N2 are that NP does not require feedback for *individual* packets but for an entire transmission group. Also NP transmits parity packets for loss recovery, while N2 retransmits the original packets that are lost.

[1] Or reset, to prevent the loss of the NAK or the retransmission.

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 throughputs 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:

$$\Lambda_o^w = min\{\Lambda_s^w, \Lambda_r^w\} \tag{9}$$

In the following, we compare the processing rates for protocols N2 and NP as a function of the number of receivers. The processing rates for N2 were computed in [18]. For protocol NP, the computation of the processing rates is given in the appendix. To obtain the results, we used the same values for the various processing times as [18] and made our own measurements for encoding and decoding with the coder of Rizzo [14] on the same hardware, a DECstation 5000/200 running ULTRIX 4.2a. In the throughput calculation we use $E[X_p] = E[Y_p] = 1000\mu$secs needed to send or receive a 2Kbyte data packet and $E[X_n] = E[Y_n] = E[Y'_n] = 500\mu$secs as the processing time to send or receive a NAK packet. We use $E[X_t] = E[Y_t] = 24\mu$secs for the timer overhead. We measured as coding constants $c_e = 700\mu$secs and the decoding constant as $c_d = 720\mu$secs for 2Kbyte packets and a symbol size of $m = 8$. The reader is referred to the appendix for definitions of the above quantities.

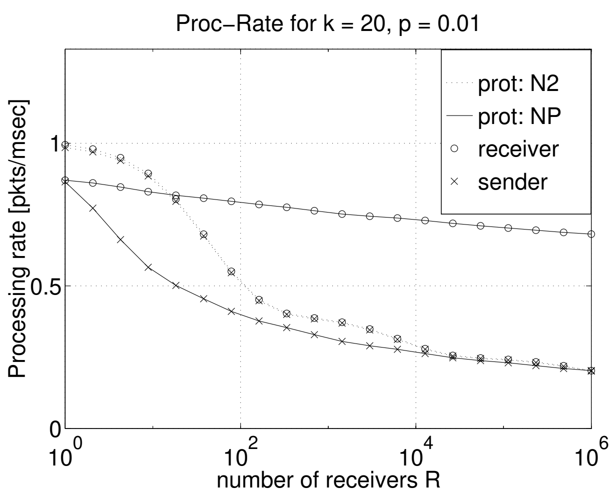

Figure 17: Processing rates at sender and receiver for protocols N2 and NP with $k = 20$ and $p = 0.01$.

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. (10) and (11)). The processing rates 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 times, both of which *depend linearly* on the mean number of transmissions (see Eq. (13) and (15)). At the receiver, the processing rate is largely determined by the decoding time, which is *independent* of the number of receivers (Eq. (16)) and the packet processing time, which increases linearly with the mean number of transmissions.

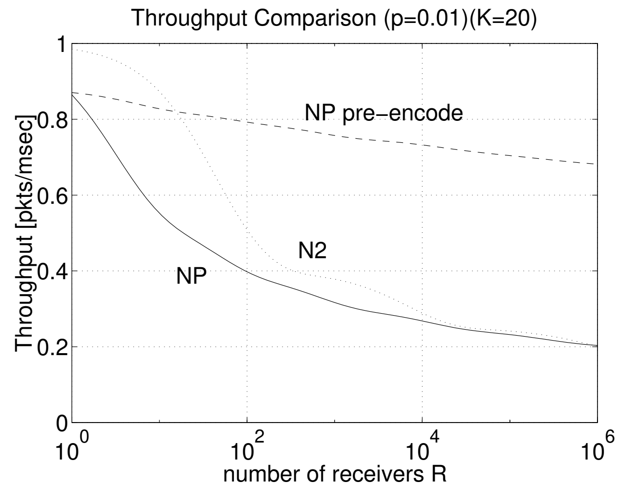

Figure 18: Throughput [pkts/msec] for N2 and for NP with and without pre-encoding for $k = 20$ and $p = 0.01$.

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

Protocol NP contains two improvements over N2: loss recovery via parity retransmission and feedback reduction due to a single NAK *per transmission round* instead of per missing packet. By slightly modifying Eq. (13) and (14) we obtained the processing rates for the case *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 did not change and the receiver processing rate decreases only slightly for a very large number of receivers.

From the processing rates obtained, we conclude that NP scales much better than N2 as the number of receivers increases: the sender becomes the bottleneck and not the receivers. If required, there are some easy solutions to match the speed of the sender and the 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 is used at the sender, or *(iii)* dedicated hardware is used for encoding. Figure 18 compares the throughput as given by Equation (9) 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 those 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 cost of coding at 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.
- The error-control feedback is reduced.
- A moderate transmission group size of $k = 20$ will tolerate burst losses, even without interleaving.
- Scalability is achieved for large numbers of receivers up to 1 million.

We can draw the following conclusions:

- Integrated FEC dramatically reduces the mean number of transmissions as compared to non–FEC.
- Integrated FEC is better than layered FEC for all parameters, low redundancy is sufficient to achieve idealized integrated FEC.
- Layered FEC can reduce the number of transmissions needed for large receiver populations to receive a packet. However, unlike integrated FEC, its performance is sensitive to the coding parameters and the presence/absence of burst losses. It may be reasonable for applications with delay constraints; this is a topic for future work.
- High loss receivers determine the performance, even if they are a small subset of all receivers.
- With shared loss the whole repair efficiency of FEC is not used in the same extent than in the case of independent loss. Shared loss can be modeled by a reduced number of receivers that lose independently.
- For burst loss layered FEC is worse than no FEC. The performance of integrated FEC decreases, when losses are bursty, especially for small values of k. For small values of k interleaving helps a bit. Integrated FEC with a large transmission group size k does not need interleaving.
- For protocols based on integrated FEC (such as NP) the sender is the bottleneck. Pre-encoding or a strong sender machine will lead to a up to 3 times higher end-system throughput compared to multicast without FEC, even when receivers decode online.

Acknowledgment

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.

References

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

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

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

[4] S. Floyd, V. Jacobson, C. Liu, S. McCanne, L, and Zhang, "A Reliable Multicast Framework for Light-weight Sessions and Application Level Framing", *Submitted to IEEE/ACM Transactions on Networking*, 1996.

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

[6] J. C. Lin and S. Paul, "RMTP: A Reliable Multicast Transport Protocol", *INFOCOMM '96*, pp. 1414–1424, San Francisco, CA, March 1996.

[7] 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.

[8] 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.

[9] 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.

[10] 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.

[11] 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)*, INRIA, Sophia Antipolis, FRANCE, October 1996, IFIP, Chapman & Hall.

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

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

[14] L. Rizzo, "Effective erasure codes for reliable computer communication protocols", *Computer Communication Review*, April 1997.

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

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

[17] 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*, Springer Verlag, Heidelberg, Germany, april 1995.

[18] 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.

[19] 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.

Appendix

Let us first recall from [18] the equations for the processing rates for protocol N2.

$$\begin{aligned}
1/\Lambda_s^{N2} &= E[X^{N2}] &(10)\\
&= E[M^{N2}]E[X_p] + (E[M^{N2}]-1)E[X_n]\\
1/\Lambda_r^{N2} &= E[Y^{N2}] &(11)\\
&= E[M^{N2}](1-p)E[Y_p]\\
&\quad +(E[M^{N2}]-1)\left(\frac{1}{R}E[Y_n] + \frac{R-1}{R}E[Y_n^{'}]\right)\\
&\quad +P[M_r>2](E[M_r|M_r>2]-2)E[Y_t]
\end{aligned}$$

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 transmission groups. 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, k and h remain as before. To simplify the analysis we make the following assumptions:

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

$$\Lambda_o^{NP} = min\{\Lambda_s^{NP}, \Lambda_r^{NP}\} \qquad (12)$$

With

$$\begin{aligned}
1/\Lambda_s^{NP} &= E[X^{NP}] &(13)\\
&= E[X_e] + E[M^{NP}]E[X_p] + \frac{E[T]-1}{k}E[X_n]\\
1/\Lambda_r^{NP} &= E[Y^{NP}] &(14)\\
&= E[M^{NP}](1-p)E[Y_p]\\
&\quad +((E[T]-1)/k)\left(\frac{1}{R}E[Y_n] + \frac{R-1}{R}E[Y_n^{'}]\right)\\
&\quad +P[T_r>2](E[T_r|T_r>2]-2)E[Y_t] + E[Y_d]
\end{aligned}$$

$E[M^{NP}]$ is computed as $E[M]$ of Eq. (6).

For the RSE coder of Rizzo we compute per packet coding and decoding as

$$\begin{aligned}
E[X_e] &= k\cdot(E[M^{NP}]-1)\cdot c_e &(15)\\
E[Y_d] &= k\cdot p\cdot c_d &(16)
\end{aligned}$$

where $k \cdot p$ denotes the mean number of data packets 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 and the the size of the packets.

The mean number of transmission rounds is

$$\begin{aligned}
E[T_r] &= \sum_{m=0}^{\infty}(1-P[T_r\le m])\\
E[T_r|T_r>2] &= \frac{E[T_r]-P[T_r=1]-2P[T_r=2]}{P[T_r\ >2)}\\
E[T] &= \sum_{m=0}^{\infty}(1-P[T\le m]) &(17)
\end{aligned}$$

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

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

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

from the expressions derived in [19], which contains the assumption 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 give an upper bound on the expected number of transmission rounds.