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

Technical Report 97-17
Department of Computer Science
University of Massachusetts
March 1997

Abstract

In this paper 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

*The work of D. Towsley was supported in part by INRIA, the Defense Advanced Research Projects Agency under contract F19628-95-C-0146, 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.

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.

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 substantially reducing the packet loss probability and, thus, reducing the number of packet retransmissions and network bandwidth requirements. If integrated with ARQ, then FEC introduces the following:

- A *single parity packet* can repair the loss of *different data packets at different receivers*.
- Using parity packets for loss repair, the sender needs to know only the maximum number of packets lost by any receiver but not their sequence numbers. Feedback from a single receiver is reduced to a single number for a group of packets.

This has the consequence of substantially reducing the network bandwidth requirements of an application requiring reliable multicast data transport. Such a scheme is also referred to in the literature as **hybrid ARQ** [LCM84].

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.

1.1 Related Work

An early paper by Metzner [Met84] 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 [Hui96] 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 [FJL$^+$96, SDW92] or introduce a hierarchy [LP96, Hof96, YGS95]. 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. In Section 2 a brief overview over FEC is given. 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 [Met84, SK95, Den93, Hui96, NB96]. 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 [McA90], is used to generate the redundant data. Suppose we have a set of k data packets $\{d_1, d_2, ..., d_k\}$ each of which is P bits long. The RSE encoder takes $\{d_1, d_2, ..., d_k\}$ and produces a set $\{p_1, p_2, ..., p_{n-k}\}$ of packets each P bits long called **parities**. We also use the parameter h to denote the number $n - k$ of parities. For the purpose of coding, we consider the data packets $\{d_1, d_2, ..., d_k\}$ as elements of the Galois field $GF(2^P)$ [LC83] and define the polynomial $F(X)$ as

$$F(X) = d_1 + d_2 X^1 + ... + d_k X^{k-1}. \tag{1}$$

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

In the case of loss, the RSE **decoder** at the receiver side can reconstruct the data packets $\{d_1, d_2, ..., d_k\}$, whenever it has received **any** k out of the n packets $\{d_1, d_2, ..., d_k, p_1, p_2, ..., p_{n-k}\}$. The k data packets will also be referred to as a **transmission group** or **TG**. The n packets $\{d_1, d_2, ..., d_k, p_1, p_2, ..., p_{n-k}\}$ 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 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 to 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: An ARQ scheme that retransmits the original packets that are lost must know the sequence numbers of the lost packets. Using parity packets for loss repair, the sender need only know the maximum number of packets lost by any receiver but not their sequence numbers. Feedback from a single receiver is reduced to feedback of a single number.

  When a lost packet is retransmitted via multicast, the packet will be received more than once by all of the receivers that have already successfully received the packet. 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 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 [McA90] uses a symbol size of m with $m = 8$ or $m = 32$. The software (SW) implementation of the same RSE coder by Feldmeier uses $m = 8$ or $m = 16$ for execution on a 32-bit processor. To obtain a fast SW implementation, the symbol size should not be larger than half the word size of the processor [Fel97]. 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 [McA90]. 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, $m = 16$ will be sufficiently large.

A software version of another RSE coder was recently put into public domain by L. Rizzo [Riz97]. In Figure 1 the coding and decoding throughput of this coder is shown. The measurements are made 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 k and h/k.

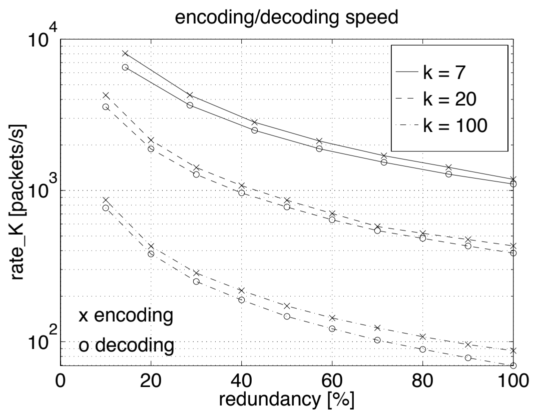


Figure 1: Coding and decoding rates in packet/s with respect to the redundancy h/k and the transmission group size k.

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 $1.25ms$. Such high performance and the fact that the bandwidth requirements of many current multicast applications are typically less than 100 KBytes, 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.

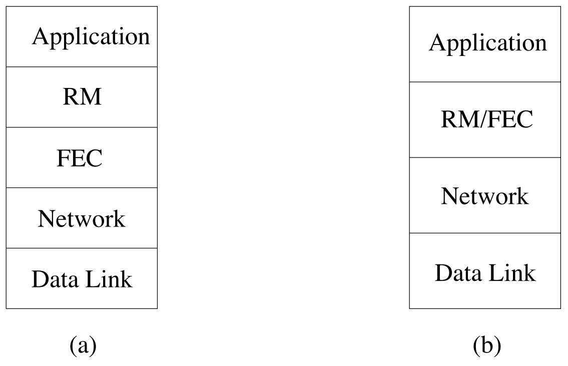


Figure 2: (a) Layered FEC. (b) Integrated FEC.

3 Placement of FEC in 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). 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 [Hui96] has established the benefits of layered FEC in terms of reducing network resource usage. 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.

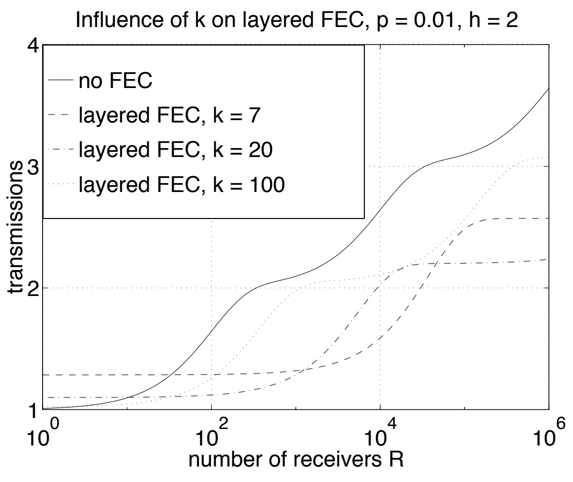
The emphasis in this section and Section 4 will be on comparing the average number of transmissions required to transmit a packet reliably to all receivers with layered FEC, integrated FEC, and no FEC. This metric is important because it reflects directly the network bandwidth required to support reliable 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 that reductions in the number of transmissions required will often lead to reductions in latencies.

Throughout this section we will consider a single sender sending to R receivers. We assume that packet losses occur as independent events, both spatially and temporally with probability p. The assumption of identical loss probabilities will be relaxed in Section 3.3 where we examine the effect that different loss probabilities has on the performance of reliable multicast.

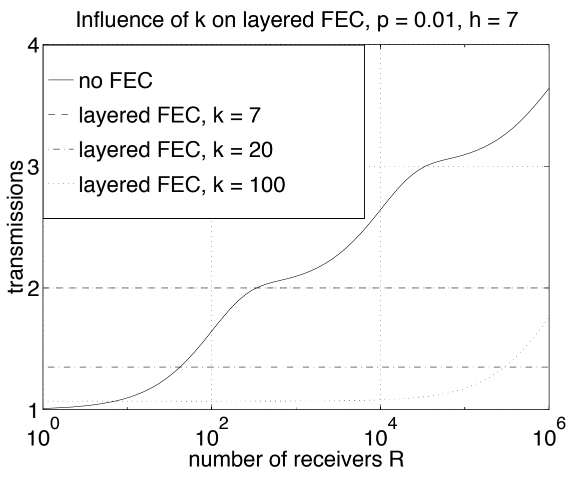
3.1 Layered FEC

Consider layered FEC based on the RSE codes described in the previous section. The sender multicasts a TG consisting of k packets followed by $n-k$ parity packets. Each receiver then recovers as many of the k packets as possible. In particular, packet i from the a TG is sent up to the RM receiver if either it is received by the FEC receiver or at least k of the other $n-1$ packets are received correctly. This was first proposed and studied in [Hui96].

Let $q(k,n,p)$ denote the probability that the FEC receiver is unable to decode a random packet. This



(a)



(b)

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

probability is given by

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

Let $E[M]$ denote the average number of packet transmissions required in order to deliver an arbitrary packet to all receivers. It is given by

$$E[M] = \sum_{i=0}^{\infty} (1 - (1 - q(k,n,p)^i)^R) n/k. \tag{3}$$

Figure 3 illustrates the typical behavior of layered FEC for different size transmission groups, $k = 7, 20, 100$ when the numbers of parity packets are 2 and 7. First, we observe a substantial decrease in the expected number of transmissions when FEC is introduced, an observation first made in [Hui96]. Second, we observe that the number of parity packets needs to be matched to the TG size. For example, a TG of size $k = 100$ 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 $a \leq h$ of the 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

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

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

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

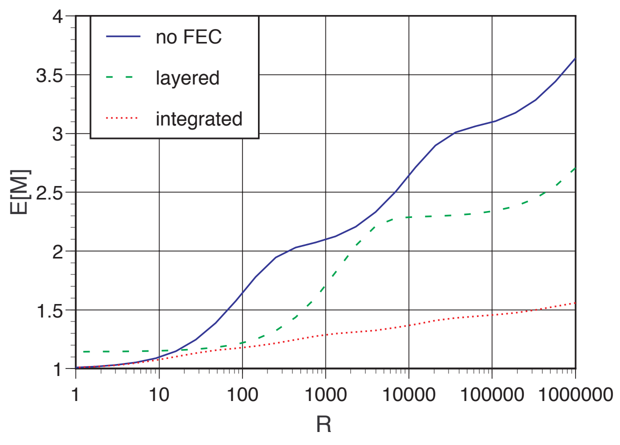
Finally, defining M as before, the mean number of transmissions per arbitrary packet is

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

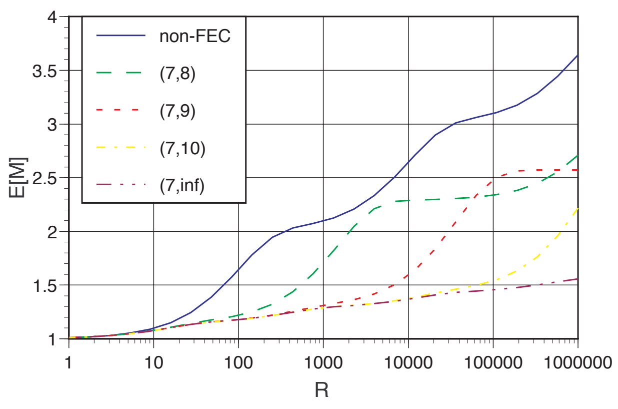
Next we derive an expression for $E[M]$ in the case that $n < \infty$. Let B denote the number of times that an arbitrary packet was transmitted, i.e., the number of blocks transmitted that included it. Note that the transmission of the first $B - 1$ groups require the transmission of n packets each. 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

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

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



(a)



(b)

Figure 4: $E[M]$ versus R for TG size of 7 and loss probability of $p = 10^{-2}$; (a) Layered FEC versus integrated FEC; (b) Integrated FEC with $k = 7$ for different values of h.

Figure 4(a) 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 4(b) 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, Figure 5(a) illustrates the influence of the TG size, k, on the performance of integrated FEC. Figure 5 shows that increasing the transmission group size reduces the number of transmissions for integrated FEC nearly down to one – there are nearly no retransmissions, even for a large number of receivers. On the other hand it can be seen that with the increase of the transmission group size from $k = 7$ to $k = 20$ and $k = 100$, only a relatively small performance gain is achieved.

We observe from Figure 5(b) that this behavior holds when the loss probability is varied – a high increase of the transmission group size does not affect the performance of integrated FEC much.

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

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

for layered FEC. In the case of integrated FEC, E[M] is as derived in Section 3.2, where heterogeneous receivers are modelled by

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

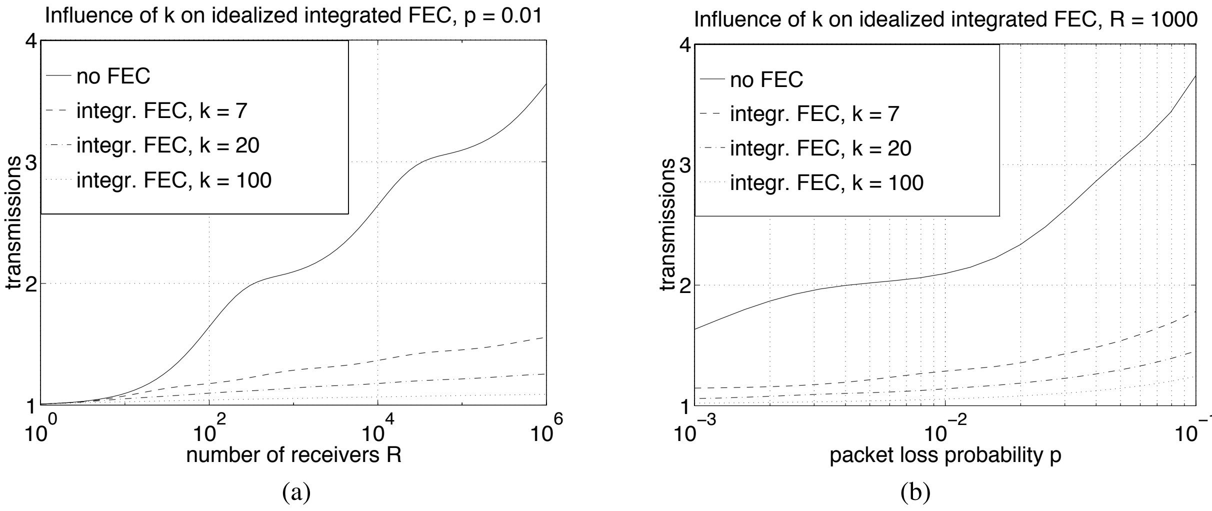


Figure 5: Integrated FEC for different TG sizes for packet loss probability $p = 10^{-2}$. (a) Influence of number of receivers; (b) Influence of number of loss probability for $R = 1000$ receivers.

where $P(L_r \leq m)$ is calculated using the packet loss probability $p(r)$ substituted in for p. We consider a simple model where the population consists 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 $\nu \times 100\%$ of high loss receivers among all receivers.

We will 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.

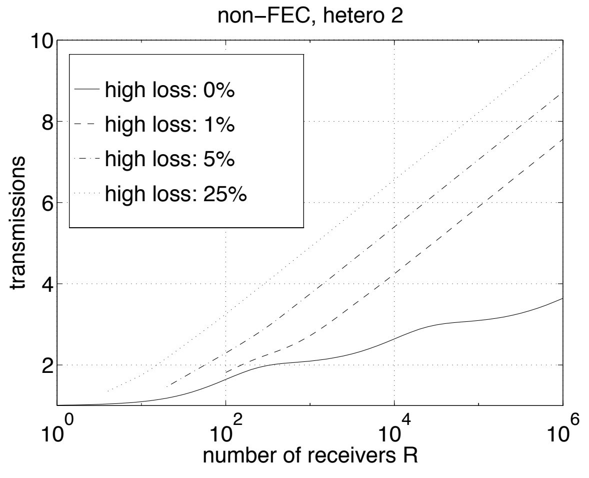
The results for reliable multicast without FEC (Figure 6(a)) and integrated FEC (Figure 6(b)) are similar. In both cases the loss probability of a standard receiver is $p = 10^{-2}$, the loss probability of a high loss receiver is $p = 0.25$.

It can be seen that the influence of the percentage of 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 (Figure 6). 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 when there is no FEC.

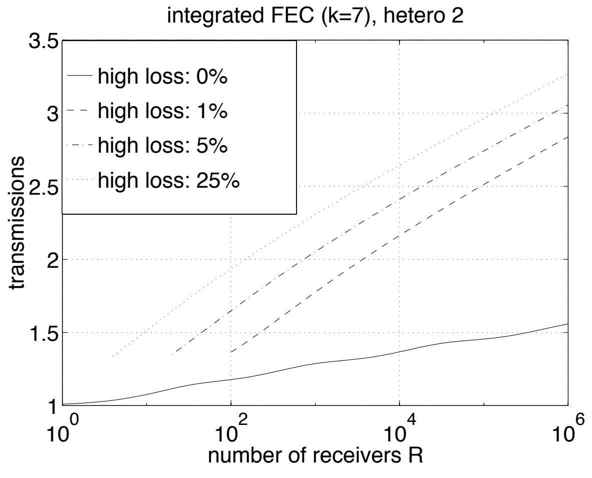
For real multicast groups the percentage of high loss receivers is in most cases determined by the position of a high loss router in the multicast tree and the number of receivers that experience the high loss of this router. In this case the receivers downstream will be equally affected by a loss –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

4 Effect of correlated loss on 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.



(a)



(b)

Figure 6: Reliable multicast for different heterogenities (a) without FEC; (b) 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. We will explore in this section whether and how the presence of such losses affect the conclusions which we have drawn from the independent loss model until now. In order to investigate the impact of shared loss we will consider the following two loss models:

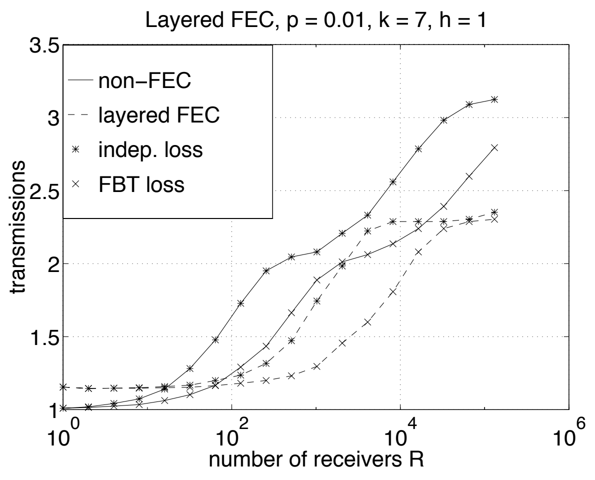
- **independent loss**: Only the receivers lose packets, each receiver independently with probability p. Other nodes of the multicast tree do not lose packets at all.
- **FBT shared loss**: We consider a full binary tree (FBT) of height d, where losses occur as independent events at each node (incl. 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 the receivers is p:

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

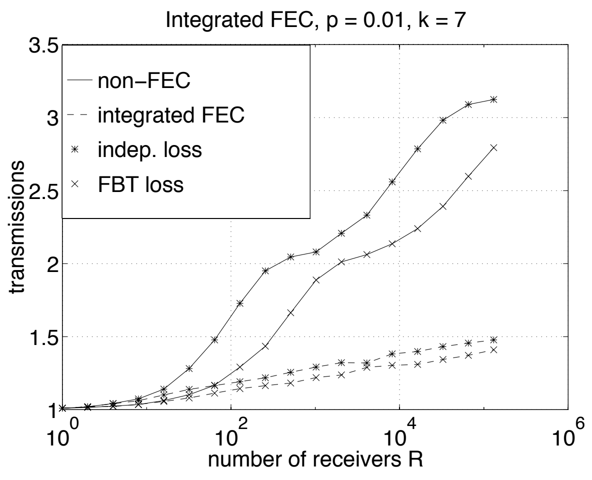
Note that each receiver experiences the same loss probability in both models and that there is no temporal loss correlation.

The derivation of the expected number of transmissions required to correctly transmit a packet reliably over a FBT was first given in [BMT94]. 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 a loss occurs or not. 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 loss and may be transmitted needlessly. Figure 7 (a) 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 modelled by a full binary tree) layered FEC is worthwhile only for multicast group sizes that are larger than those that the analysis for independent loss suggests.



(a)



(b)

Figure 7: FEC compared with non-FEC for Independent and for FBT Shared Loss: (a) Layered FEC (b) Integrated FEC.

The impact of shared loss on FEC is shown in Figure 7(a) and (b) for layered FEC and integrated FEC respectively. 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.

Another useful observation to make 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 based on measurements of receiver loss rates will overestimate the number of transmissions of reliable multicast, when loss is modelled as independent. For FEC this means that adapting the amount of redundancy due to measured loss rates at the receivers can lead to an overestimate of the amount of redundancy needed – when independent loss is assumed.

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 $X_t \in \{0, 1\}$. If a packet is transferred at time t, then no losses occur if $X_t = 0$. On the other hand, the packet is lost if $X_t = 1$. 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, $p_{i,j}(t) = P(X_{\tau+t} = j \mid X_\tau = i)$. It is given by (see [Mor58, ch. 6] for details).

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

for all $t > 0$.

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

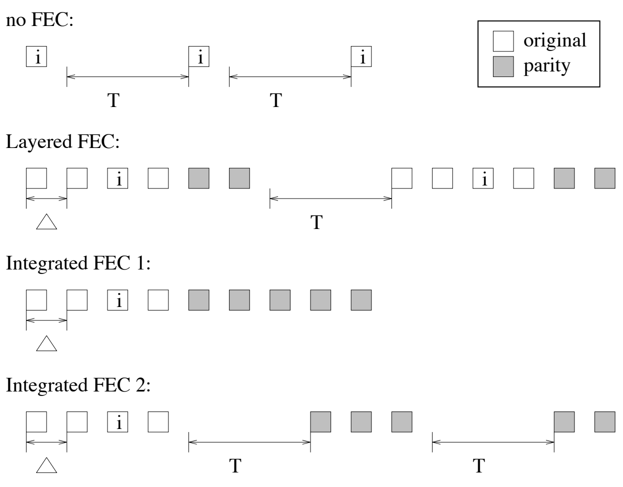
When burst losses occur, the timing of the retransmissions has an influence on the performance of the loss recovery. To further investigate this point, we consider different cases as shown in figure 8.

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

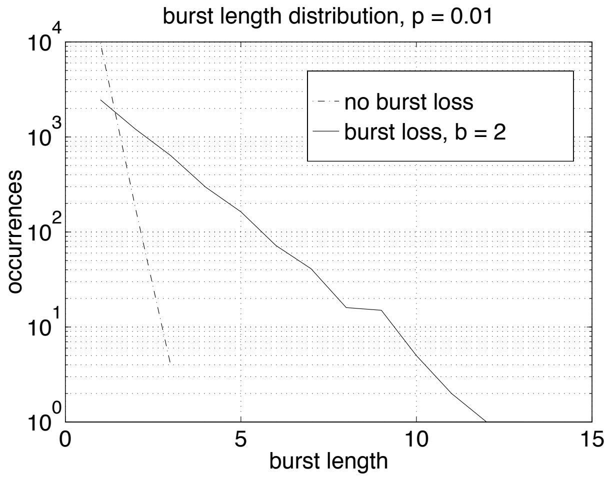
For integrated FEC, the timing considerations depend on the protocol that implements loss recovery by parity transmissions. We distinguish between two cases:

- **Integrated FEC 1**: The transmissions of packets, data and parity, within the same block are spaced Δ time units apart. 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 interarrival 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.

The parameter T may correspond to the maximum roundtrip time that the application encounters. However, it can also be chosen so that packet/TG retransmissions under no FEC, layered FEC, and integrated FEC 2 will suffer ransom rather then bursty losses. Note that the gaps of size T in these schemes can be used to transmit other packets or TGs. This is similar to the concept of *interleaving* commonly used in



(a)



(b)

Figure 8: (a) Timing considerations of the different approaches. (b) Distribution of number of consecutive losses at one receiver, for no burst and burst ($\bar{b} = 2$) for a packet loss probability of $p = 0.01$.

communications in order to convert a channel characterized by bursty errors to one characterized by random errors.

Let $\lambda = 1/\Delta$ be the packets transmission rate and let $\bar{b}$ be the expected number of consecutively lost packets. Let X denote the number of consecutive subsequent packets that find the loss model in state loss. The probability that the model is in state loss for $i \geq 1$ packets is:

$$P(X = i) = \pi_1 p_{1,1}(\Delta)^{i-2} p_{1,0}(\Delta)$$

The distribution function is:

$$P(X \leq i) = 1 - \pi_1 p_{1,1}(\Delta)^{i-1}$$

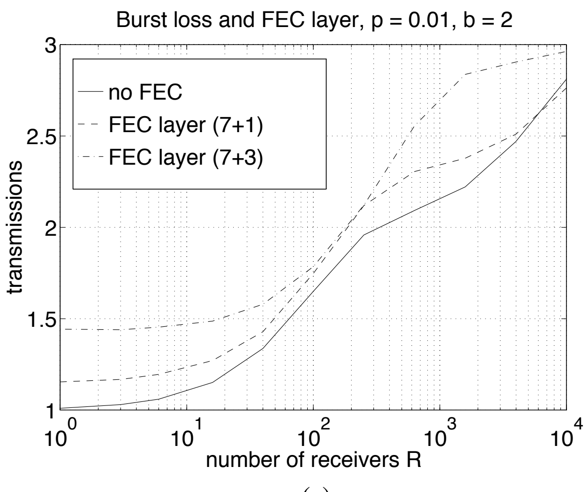
and the expected number of lost packets in sequence (burst loss length)is:

$$\bar{b} = E(X) = \sum_{i=1}^{\infty} 1 - P(X \leq i) = \pi_1 \frac{1}{1 - p_{1,1}(\Delta)} \tag{9}$$

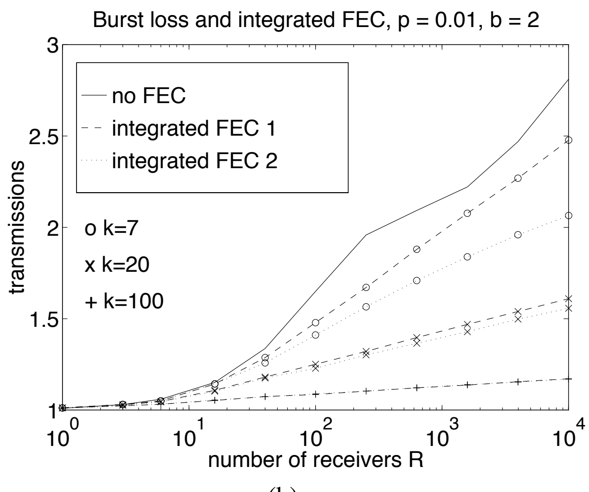
Given the packet loss probability p, the average burst loss length $\bar{b}$ in packets and the sending rate λ, the parameters for the loss model are:

$$\begin{aligned} \mu_0 &= -\pi_1 \lambda \log(1 - \frac{1}{\bar{b}}) \\ \mu_1 &= \mu_0 \frac{1-p}{p} \end{aligned}$$

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 [Bol95] for a loaded IP path between Sophia Antipolis (INRIA) and London (UCL). The packet loss probability was chosen to be $p = 0.01$. Last, T is chosen to be 300 ms. Figure 8(b) illustrates the burst length distribution at one receiver for independent and for burst loss among packets for these parameters. It can be seen that the occurrences decrease linearly on a logarithmic scale.



(a)



(b)

Figure 9: (a) 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 $\bar{b} = 2$. (b) Comparison of integrated FEC 1 and integrated FEC 2 for transmission group sizes $k = 7, 20, 100, p = 0.01$ and $\bar{b} = 2$.

We observe from Figure 9(a) 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.

Although large TGs are not desirable under layered FEC, it is reasonable to consider large TGs under integrated FEC. Figure 9(b) 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 outperforms integrated FEC 1 for $k = 7$, since parity packets belonging to the same transmission group are spread out over time (see Figure 8(a)) and are more likely to span across a loss period. Figure 9(b) 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 TG spans a sufficiently long enough 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 reliable multicast integrated 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 to a generic reliable multicast protocol without FEC.

5.1 Protocol NP

There are obviously numerous ways to design a reliable multicast protocol with hybrid ARQ. The design choices are largely influenced by considerations such as the type of application and its constraints.

The protocol NP uses integrated idealized FEC 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 of k data packets. For each TG, $n-k$ parity packets are generated.

For feedback from the receivers to the sender, multicast transmission of NAKs as in SRM [FJL$^+$96] is used and NAK suppression is performed.

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

- Round 1: The k data packets of TG_i are sent
- Round 2: $l^{(1)}$ parities for TG_i are sent, where $l^{(1)}$ is the maximum number of packets lost in round 1 for TG_i .
- Round j: $l^{(j-1)}$ parities for TG_i are sent, where $l^{(j-1)}$ is the maximum number of packets lost in round $j-1$ for TG_i .

The Sender:

- Transmits the k data packets in transmission group TG_i.
- When done, the sender polls the receivers with `POLL(i,k)` 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 will then resume transmission of the interrupted transmission group TG_m.

The Receiver:

- Data packets and parities for TG_i are stored 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 throughput) into account.
- If a `NAK(i,m)` is received with $m \geq l$ the timer for `NAK(i,l)` is canceled[1]. When the timeout for `NAK(i,l)` occurs, `NAK(i,l)` is sent.

Our slotting and damping mechanism assures that the sender will 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 [TKP97]: feedback is receiver-initiated, NAKs are sent via multicast. A receiver receiving a NAK will not generate a NAK for the same round. The

[1]Or reset, to allow for the retransmission to take place.

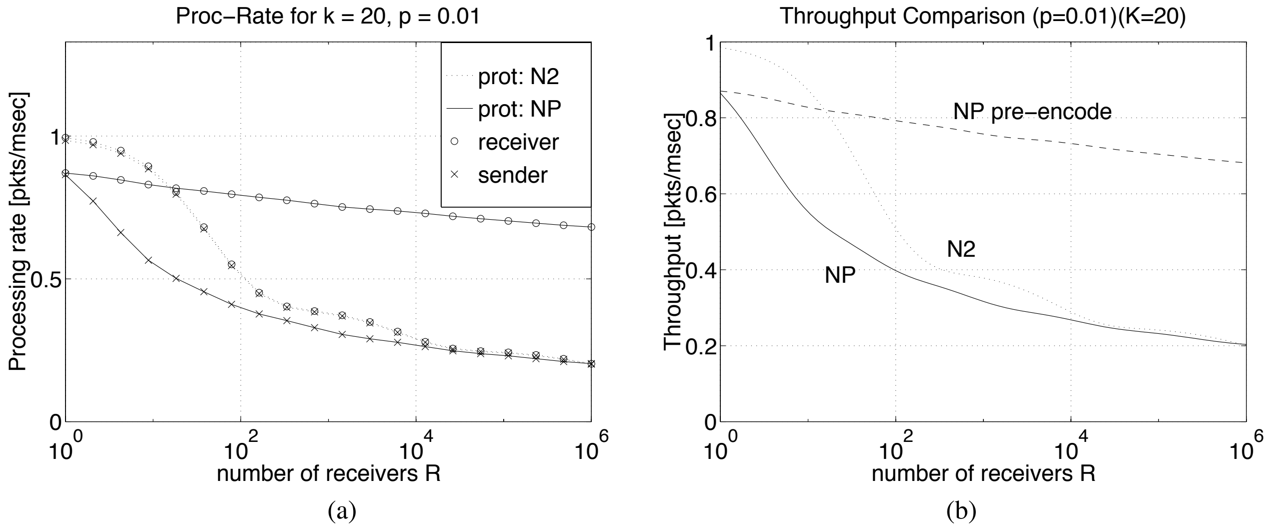


Figure 10: (a) Processing rates at sender and receiver for protocols N2 and NP with $k = 20$ and $p = 0.01$. (b) Throughput [pkts/msec] for N2 and for NP with and without pre-encoding for $k = 20$ and $p = 0.01$.

major differences between NP and N2 are that NP does not require feedback on *individual* packets but on the reception status of an entire transmission group. Also 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 throughputs for the case of a one-to-many transmission. Let Λ_s^w, Λ_r^w be the send and receive per packet processing rates of protocol $w \in \{N2, NP\}$. Then the achievable endsystem throughput Λ_o^w is defined as the minimum of sender and receiver processing rate:

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

In the following, we compare the processing rates for the protocol N2 and NP as a function of the number of receivers. The processing rates for N2 were computed in [TKP97]. 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 [TKP97] and made our own measurements for encoding and decoding with the coder of Rizzo [Riz97] 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 8. The reader is referred to the appendix for definitions of the above expectations.

In Figure 10(a), we see that the sender and receiver processing rates are nearly identical for 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)). The decrease in the processing rates as the number of receivers increases is 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 Eqs. (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 an average 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 need only decode (reconstruct) 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 the transmission of a *single NAK per transmission round* instead of per missing packet. By slightly modifying Eq. (14) and (15) we obtained the processing rates for the case of *one NAK 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: for the sender, the rate did not change at all, for the receiver, the rate decreases only slightly for 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 offline 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 10(b) compares the throughput as given by Equation (10) for N2 and NP with and without pre-encoding, for $k = 20, p = 0.01$. It demonstrates to which extent encoding influences the performance of the NP protocol. Anyway, it can be seen that already for a very low number of receivers the throughput of NP with pre-encoding is higher than for N2 and NP without pre-encoding. While these results indicate that software coding/decoding is fast enough *today*, this will be even more the case in the future, since processing rates tend to increase faster than transmission bandwidth [Fra91].

6 Summary

Using FEC in an integrated fashion provides five major benefits:

- Integrated FEC shifts resource usage from the network to the endsystems: The number of transmissions is reduced and therefore the network bandwidth used as well – at the cost of coding at the endsystems.
- The achievable endsystem 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.
- 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 modelled by a reduced number of receivers that lose independently.

- For burst loss is layered FEC worse than no FEC. Integrated FEC is getting much worse also, especially for small values of k. For small values of k interleaving helps a bit. For large values of k interleaving is not needed, as integrated FEC achieves the same repair efficiency than without burst loss.
- For protocols based on integrated FEC (NP) is the sender the bottleneck. Pre-encoding or a strong sender machine will lead to a up to 3 times higher endsystem throughput compared to multicast without FEC, even when receivers decode online.

Acknowledgements: The authors are grateful to Stefan Roosli for measuring the performance of the FEC coder and decoder, to Jean Bolot for his suggestions on the burst loss model and to Brian DeCleene for discussions which led to our comparing the layered and integrated approaches.

Appendix

Let us first recall from [TKP97] the equations for the processing rates for protocol N2.

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

and

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

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 the receiver
Y_d	- time to decode a packet at the 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 the receiver
$Y_n^{'}$	- time to receive and process a NAK at the 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.

$$\Lambda_o^{NP} = min\{\Lambda_s^{NP}, \Lambda_r^{NP}\} \tag{13}$$

With

$$1/\Lambda_s^{NP} = E[X^{NP}] = E[X_e] + E[M^{NP}]E[X_p] + ((E[T]-1)/k)E[X_n] \tag{14}$$

and

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

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

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

$$E[X_e] = k \cdot (E[M^{NP}] - 1) \cdot c_e \tag{16}$$

$$E[Y_d] = k \cdot p \cdot c_d \tag{17}$$

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

$$E[T_r] = \sum_{m=0}^{\infty}(1 - P[T_r \le m]) \tag{18}$$

$$E[T_r|T_r > 2] = (E[T_r] - P[T_r = 1] - 2P[T_r = 2])/P[T_r \ > 2) \tag{19}$$

$$E[T] = \sum_{m=0}^{\infty}(1 - P[T \le m]) \tag{20}$$

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 [AGO93], 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 max. number of parities required by any receiver, this assumption will give an upper bound on the expected number of transmission rounds.

References

[AGO93] 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.

[BMT94] Pravin Bhagwat, Partho P. Mishra, and Satish K. Tripathi. Effect of topology on performance of reliable multicast communication. In *Proceedings of INFOCOM'94*, volume 2, pages 602–609, Toronto, Ontario, Canada, June 1994. IEEE.

[Bol95] J. C. Bolot. Analysis and control of audio packet loss in the internet. In T. D. C. Little and R. Gusella, editors, *5th Workshop on Network and Operating System Support for Digital Audio and Video*, volume 1018 of *LNCS*. Springer Verlag, Heidelberg, Germany, april 1995.

[Den93] 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.

[Fel97] David C Feldmeier. Personal communication. January 1997.

[FJL$^+$96] 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.

[Fra91] A. G. Fraser. Designing a public data network. *Communications*, 29(10):31–35, October 1991.

[Hof96] M. Hofmann. A generic concept for large-scale multicast. In B. Plattner, editor, *Proc. International Zuerich Seminar*, volume 1044 of *LNCS*, pages 95–106. Springer Verlag, February 1996.

[Hui96] Christian Huitema. The case for packet level fec. In *Proceedings of IFIP 5th International Workshop on Protocols for High Speed Networks (PfHSN'96)*, INRIA, Sophia Antipolis, FRANCE, October 1996. IFIP, Chapman & Hall.

[LC83] S. Lin and D. J. Costello. *Error Correcting Coding: Fundamentals and Applications*. Prentice Hall, Englewood Cliffs, NJ, 1983.

[LCM84] S. Lin, D. J. Costello, and M. J. Miller. Automatic-repeat-request error-control schemes. *IEEE Commun. Magazine*, 22(12):5–17, 1984.

[LP96] John C. Lin and Sanjoy Paul. Rmtp: A reliable multicast transport protocol. In *INFOCOMM '96*, pages 1414–1424, San Francisco, CA, March 1996.

[McA90] A. J. McAuley. Reliable broadband communications using a burst erasure correcting code. In *Proc. ACM SIGCOMM 90*, pages 287–306. Philadelphia, PA, September 1990.

[Met84] J. Metzner. An improved broadcast retransmission protocol. *IEEE Transactions on Communications*, COM-32(6):679–683, June 1984.

[Mor58] P.M. Morse. *Queues, Inventories, and Maintenance*. John Wiley, 1958.

[NB96] J. Nonnenmacher and E. W. Biersack. Reliable multicast: Where to use forward error correction. In W. Dabbous and Chr. Diot, editors, *Proc. 5th Workshop on Protocols for High Speed Networks*, pages 134–148, Sophia Antipolis, France, October 1996. Chapman & Hall.

[Riz97] L. Rizzo. On the feasibility of software fec. Technical report, Univ. di Pisa, Italy, January 1997.

[SK95] 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.

[SDW92] W.T. Strayer, B.J. Dempsey, A.G. Dempsey. *XTP: The Xpress Transfer Protocol*, Addison-Wesley, 1992.

[TKP97] 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*, 1997.

[YGS95] Rajendra Yavatkar, James Griffoen, and Madhu Sudan. A reliable dissemination protocol for interactive collaborative applications. In *Proceedings of ACM Multimedia*, pages 333–344, San Francisco, CA USA, 1995. ACM.