Packet Recovery in High-Speed Networks Using Coding and Buffer Management*

Nachum Shacham
Paul McKenney
Information and Telecommunication
Sciences Center SRI International
Menlo Park, CA 94025

Abstract

Traditional data reliability techniques such as retransmissions can result in intolerable storage requirements and data delay when they are used in gigabit wide-area networks. This paper presents a novel technique based on forward error correction (FEC) that allows the destination to reconstruct missing data packets by using redundant parity packets that the source adds to each block of data packets. Methods for generating several types of parity packets are described, along with decoding techniques and their implementations. We also present algorithms for packet interleaving and selective rejection of packets from node buffers, both of which disperse missing packets among many blocks, thereby reducing the required coding complexity. Performance evaluation, by both analytic and simulation models, shows that this technique can result in a reduction of up to three orders of magnitude in the packet loss rate.

1 INTRODUCTION

Rapid progress in the development of fiber optics and components for photonic transmission, reception, coupling, filtering, and related communications functions has created the technology for constructing gigabit/second multi-user networks [13]. So far most of the research in high speed communication systems has focused on local and metropolitan area networks (LAN and MAN) [6, 14, 16, 15, 12], and on high-speed switches [10, 17, 1]. Efforts are now under way to design and construct wide-area networks to span large geographical regions and provide long-distance data paths of bandwidth in excess of 1 gigabit/second to individual end users. We call such systems gigabit, wide-area networks (GWANs).

GWAN data paths are composed of fiber-optic links with very low bit-error rates (BER). It is not uncommon to achieve error rates of 10^{-14} over a long-distance fiber cable, despite the presence of regenerators every 100 km.

Despite their enormous link bandwidths, GWANs are still expected to experience bottlenecks due to limited storage, processing power, and switching speed. Statistical resource sharing by a large number of high speed, fluctuating-rate streams is bound to cause congestion and occasional buffer overflow. Even when a GWAN guarantees users a minimum data rate, its flow-enforcement mechanisms, such as Leaky Bucket [17], are designed to delete packets that exceed the network's ability to switch and forward. Packet loss is likely to be the dominant factor in data distortion in GWANs.

Network switches and end users traditionally rely on acknowledgement (ACK)-based closed-loop control to recover lost packets, and adjust incoming traffic rates to overcome congestion [3]. The high delay·bandwidth product of long distance data paths in GWAN results in a large "path storage." For example, assuming data travel at the speed of light, the propagation delay on a 3000 km path is 10 ms, which translates to 10^7 bit durations at 1 gigabit/second. In such an environment, ACK-based end-to-end protocols require large data storage, reduce channel utilization, and may cause instability. Because of these reasons, end-to-end protocols based on *open-loop* control are being developed for GWAN. An example of this trend is the aforementioned Leaky Bucket flow control.

Another example of open-loop control is forward error correction (FEC) coding, in which a recipient can recover lost or erroneously received data based on parity bits, which are added by the source to the information sequence. Recovering lost packets reduces the need for retransmissions of reliable data, and enhances the quality of real-time data that cannot rely on ACKs and retransmissions because of the large delays involved.

A coding scheme for GWAN must (1) deal with long sequences of lost bits (that is, missing packets), (2) correct occasional bit errors, and (3) result in low probability of undetected errors. Since bits are transmitted at extremely high rates, coding should be done with minimal processing per packet, preferably by low-complexity hardware. It should add little delay to the data and require minimal amounts of storage. In section 2 we present several coding schemes suitable for a GWAN environment. These schemes are based on grouping the data packets into blocks, to which control packets are added. Each coding scheme is limited in the number of packets it can recover; no packets may be recovered if more than that number is missing from a single block. In section 3 we discuss techniques for reducing the effect of bursts of missing packets by interleaving and by intelligent buffer management, whereby packets are selected for deletion based on their block affiliation. In Section 4 we use the reduction in packet loss rate as the performance measures for our coding schemes, and discuss the effect of block size, packet arrival rate, and amount of coding overhead on performance. We provide results from both a simple analytic model and more realistic simulations which show the limitations of the coding schemes and the conditions under which they result in a substantial reduction in loss rate. Section 5 contains some concluding remarks.

2 CODING FOR PACKET RECOVERY

2.1 General considerations

When noise-induced bit errors are the main cause for data distortion, as is the case in current computer communication networks, data are carried in the same packet as the error-control bits protecting them. However, recovery of lost packets requires that the error-control bits and the data they protect be carried in separate packets. Chiou and Li [4] proposed duplication of each data packet to enhance data survivability in high-loss military data networks. In GWAN, packet loss rate is expected to be sufficiently small to make this 1/2 rate repetition code quite inefficient.

Our approach utilizes sequence numbers, which are already required by many protocols. Since packets are sent in increasing order, a data recipient identifies missing packets by gaps in the arriving sequence, either upon the arrival of a packet with a larger number, or after a pre-determined waiting time. Thus, a missing packet can be considered as a sequence of bit *erasures*,

*This work was supported by SRI International under IR&D funds, and by DARPA under NASA contract NAS2-11555.

i.e., unreliable bits whose location is known [2]. There are several FEC codes that efficiently correct erasures, most notably the Reed-Solomon codes [2]. However, the special case we consider, in which erasures come in packet-size sequences, allows us to employ simpler techniques for recovering lost data.

The rate at which packets are lost often depends on the rate at which packets are emitted by the source. In particular, packets are more likely to be rejected because of buffer overflow as their rate of arrival to the buffer increases. Since coding increases the volume of traffic entering the network, the data packet loss rate is likely to increase after the control bits are added. Thus, the lower the code overhead (higher code *rate*), the lower is the increase in the packet loss rate. However, using control bits to recover lost packets should bring the packet loss rate below the loss rate when no coding is used. The ratio of packet loss rate after decoding to the rate when no coding is used is denoted as the *loss ratio*, and it must be smaller than 1 for the code to be useful.

In addition to the above consideration, the coding scheme must be suitable for implementation in gigabit/second networks, that is, it should require little processing and even be amenable to hardware implementation. Both coding and decoding should add a minimal amount of delay to the packets, and should require only small data storage for operation.

Based on the above considerations we propose to group data packet into *blocks* of predetermined size, and add to each block a number of *parity* packets, to contain the error-control bits. The number of parity packets, and their construction, determines the maximum number of data packets that can be recovered. However, any subset of missing packets can be recovered by using the parity packets and the other data packets of the block. The balance of this section presents several methods for constructing parity packets and using them recover lost packets. For simplicity of presentation we assume that all packets are m bits long; however, the schemes can also work with variable length packets. We begin with the simplest technique for recovering a single packet per block.

2.2 A single packet recovery

To each block of K data packets the source adds an m-bit parity packet, whose i-th bit is given by:

$$c_{K+1,i} = \left(\sum_{j=1}^{K} c_{j,i} \right) \bmod (2) \quad , \qquad i = 1, 2, \ldots, m \quad , \qquad (1)$$

where $c_{j,i}$ is the i-th bit of the j-th packet. We denote such packet as a "vertical" parity packet since, arranging the block as a rectangular array with packets as rows, $c_{K+1,i}$ is the sum modulo 2 of the bits in the i-th column.

The parity packet is generated by an encoder consisting of m exclusive-or gates (XOR, or symbolically $\oplus$), each of which has its output connected to one of its input ports, as shown in Figure 1. Packets are given, one at a time, to the encoder, with bits $1, 2, \ldots, m$ applied to the input ports of gates $1, 2, \ldots, m$, respectively. At a clock pulse, the decoder output port i represents the sum modulo 2 of bits in position i of the previous packets in the block and bit i of the current packet. There is no need to store the whole block of packets at the source; thus, immediately following its "contribution" to the encoder, a packet can be sent over the network. Following the application of the K-th packet, the gates' m output ports contain the parity packet, which is then transmitted.

The recipient of the packet sequence employs a decoder similar in structure to the encoder. For a given block, each arriving packet is applied in parallel to the m input ports of the decoder. Following the application of any subset of K packets from a block, the m output ports contain the remaining packet. If no data packet was lost or excessively delayed in the network, those first

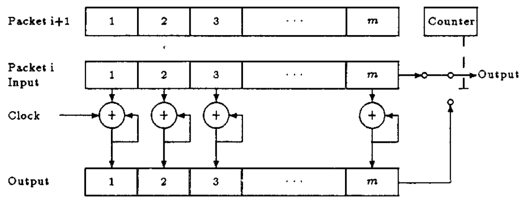

Figure 1: Parity packet generation

K packets are the data packets of the block, in which case the recipient may ignore the parity packet. If, however, data packets are missing from a block, the recipient identifies the location of the missing packets from the gap in the sequence numbers, and considers these packets as erasures. If only one of the block's data packets is missing, the K packets applied to the decoder include $K-1$ data packets and the vertical parity packet. In this case, the erasure is replaced by the contents of the parity generator's output after those K packets are applied to the decoder.

Notice that the arriving packets need not be delayed at the receiver. Every packet is applied to the parity encoder/decoder, and immediately thereafter can be forwarded onward. Furthermore, since the XOR operation is commutative, packets may be applied to the decoder in an arbitrary order. This is particularly important for networks that do not guarantee sequenced delivery.

2.3 Multiple packet recovery

When a block protected by a single vertical parity packet is received with two or more packets missing, none of those packets can be recovered. More control data must be added to recover multiple packets in a block. Since recovering an erased packet amounts to solving an equation with one unknown, there must be at least as many parity packets as there are missing packets. Each of those parity packets must add a linearly independent equation to the set.

To recover two packets in a block we propose to add a *diagonal* parity packet, the bits of which are the modulo-2 sums of bits along block diagonals, as follows:

$$c_{K+2,i} = \begin{cases} \left(\sum_{j=1}^{i} c_{j,i+1-j}\right) \bmod (2) & 1 \leq i \leq K+1 \\ \left(\sum_{j=1}^{K+1} c_{j,i+1-j}\right) \bmod (2) & K+2 \leq i \leq m \\ \left(\sum_{j=1}^{K+m-i+1} c_{i-m+j,m+1-j}\right) \bmod (2) & \\ & m+1 \leq i \leq m+K \end{cases} \qquad (2)$$

where we assume $K < m-1$. Similar equations can be written for $K \geq m-1$. Notice that the diagonal parity packet has $m+K$ bits, compared to m bits in a data or vertical parity packet. If the network requires all transmission units to be of the same size, such as ATM cells [8], a diagonal parity packet may have to occupy more than one unit. For $K < m$, a diagonal packet occupies two transmission units.

The encoder to construct the diagonal parity packet is similar in structure to that shown in Figure 1, except that its registers have $m+K$ storage elements each and the ith packet of the block ($i = 1, \ldots, K+1$) is shifted to the right and placed with its first bit in the $i-1$ element. To create a block with two-packet recovery capability, the source employs a vertical encoder as described in Section 2.2 and a diagonal encoder. The operation is still sequential, in that as soon as a packet is clocked into both encoders (possibly in parallel), it is shipped to the network. Following the clocking of the K-th packet, the vertical parity packet is ready, but before it is sent, it is also clocked into the diagonal encoder. At that point, the diagonal packet is ready.

The recipient employs two encoders identical to those used by the source. Its first step in decoding is to generate two new parity packets, one from each encoder, as follows. The data packets of a given block are clocked into each encoder and, as soon as the last one is clocked, the outputs of each encoder are summed modulo 2 bitwise with the corresponding parity packets which arrive from the networks. The new parity packets, one vertical with m and the other diagonal with $m+K$ bits, are used by the recipient to construct up to two missing packets in the block. We denote the bits of these new parity packets $d_{K+1,i}$ $(i = 1, 2, \ldots, m)$ and $d_{K+2,i}$ $(i = 1, 2, \ldots, m+K)$, respectively.

Let us examine how the recipient reconstructs two packets in a block by using the new parity packets. Assume that packets 1 and 2 are missing. The first bit of packet 1, $c_{1,1}$ can be recovered since it equals the first bit of the diagonal packet, $c_{K+2,1}$, and $c_{2,1} = c_{1,1} \oplus d_{K+1,1}$. For $1 < i \leq m$, then $c_{1,i} = c_{2,i-1} \oplus d_{K+2,i}$ and $c_{2,i} = c_{1,i} \oplus d_{K+1,i}$. Decoding of any other pair of data packets is done in a similar fashion. See Figure 2 for a possible hardware implementation of the decoder; note that the interconnections of gates and registers in that figure are for decoding packets 1 and 2, and would be somewhat different for other packet pairs.

If one data packet and one parity packet are missing from an arriving block, the other parity packet is used to recover the data packet. If the only missing packets are the parity packets, the recipient ignores them and forwards only the data packets. If there are more than two missing packets in a block, none can be recovered with this 2-packet coding scheme.

It is possible to incrementally enhance the packet recovering capability by adding parity packets which are constructed by a set of modulo-2 additions, linearly independent of the previously constructed parity packets. For example, to be able to reconstruct three lost packets in a block, we add to the previously described vertical and diagonal parity packets another diagonal parity packet, this one with parity operations along the block diagonals orthogonal to those used in the first diagonal parity packet. The source must now employ an additional encoder and the decoding scheme is somewhat more complex.

2.4 Handling bit errors

Although bit errors are rare in GWAN, they nevertheless do occur and must be anticipated. A parity bit added to a packet ("horizontal" parity), detects a single bit error in that packet, through a mismatch between its value and the modulo 2 sum of the received data bits. If a single bit error occurs in a block in which no packet is missing, the bit in the vertical parity, which is in the same position in the packet as the erroneous bit, shows a similar mismatch. The intersection of the column and row identifies the bit in error, thereby allowing the recipient to correct it. To detect more than one bit error in a packet, a stronger error detecting code is needed, for example 16-bit CRC [2]. Multiple errors per packet can be corrected by the CRC and the parity packet, provided no packet is missing from that block and no other packets contain errors. This condition is quite reasonable because of the very low error probability in fiber optic links.

If there are both missing packets and bit errors in a block, and the total number of those packets is not larger than the maximum that can be recovered, the recipient considers the packets with bit errors as erasures and recovers all the packets as described in the previous subsection.

It is also possible to add a small amount of parity bits that allow the recipient to correct bit errors in a packet and recover the maximum number of erased packet allowed by the parity packets. Consider for example a block of K data packets to which vertical and diagonal parity packets, as described in the previous subsection, are added. Suppose that in such a block packets 1 and 2 are missing and in packet 3 bit $c_{3,i}$ is erroneous.

The recipient first attempts to recover packets 1 and 2, as if packet 3 does not include a bit error. This results in an error pattern consisting of two bit errors in packet 2 and a single error in packet 1. To correct these errors the recipient needs additional $m/2$ parity bits. Each such bit includes in its construction only one of the errors in the above pattern. By knowing the packets in which the error occurred, two diagonal parity packets are sufficient to determine the location of the errors. Notice that these $m/2$ parity bits can be accommodated in the second part of the diagonal parity packet, where $K < m/2$, so that no additional packets are generated.

2.5 Recovering a burst of lost packet

Arranging the data in a matrix, and adding parity bits both for the rows and the columns, can used for correcting bit errors [5]; this idea also be employed to recover bursts of lost packets. To do so, we arrange the data packets in a $K \times M$ array and add to each row and each column of the array a parity packet. The i-th bit of a parity packet is the modulo 2 sum of the i-th bits of the data packets in the corresponding row or column. That is, if we denote by $c_{i,j,k}$ the k-th bit of the packet in the (i, j) position in the array, then:

$$c_{i,M+1,k} = \left(\sum_{j=1}^{M} c_{i,j,k} \right) \bmod (2) \qquad 1 \leq k \leq m, \;\; 1 \leq i \leq K \quad , \tag{3}$$

and

$$c_{K+1,j,k} = \left(\sum_{i=1}^{K} c_{i,j,k} \right) \bmod (2) \qquad 1 \leq k \leq m, \;\; 1 \leq j \leq M \quad . \tag{4}$$

The code rate, $\frac{K+M}{KM}$, can be varied by adjusting the dimensions of the array.

If the packets are sent by rows, this scheme can recover bursts of missing packets of any length less than or equal to M. Such recovery is done with the aid of the column parity packets only. The row parity packet are used to recover additional missing packets scattered over the array. This technique requires larger hardware complexity: $K + M$ coders similar to those described in subsection 2.2. Note that packets do not encounter larger delays at the source than they would with no interleaving. Also, if a row suffers only a single missing packet, that packet can be recovered after the last packet of the row is received. On the other hand, if two or more packets are missing in a row, they must wait until their respective columns are fully received.

3 BUFFER MANAGEMENT AND INTERLEAVING

Bit errors are caused by random processes, and although a network designer can affect the average error rate, say by adjusting signal power, he cannot select the bits that will be received in error. In contrast, a congested node selects a particular packet for deletion from the set of packets available in its buffer, according to pre programmed buffer management rules. In current networks, these rules are designed to support congestion control and improve throughput and fairness in service. For example, a buffer management scheme may give priority to packets that are already in the buffer and reject all packets that arrive when the buffer is full. Other rules may assign priority to packets based on their source or destination, on the elapsed time they have spent in the network, or according to the rate at which those packets are emitted by their source.

Buffer management rules may not affect the average rate at which packets are lost, but they can have a strong effect on the distribution of lost packets. The performance of the error control scheme described above strongly depends on this distribution. For example, two missing packets in a block are not recoverable but two packets in adjacent blocks are. Thus, the role of buffer management procedures in enhancing end-to-end error control can be that of dispersing the deleted packets so as to minimize the number of blocks that arrive with multiple missing packets. This can be done in the following manner. When an arriving packet finds a full buffer and that packet's block has already suffered a lost packet, the server deletes from its buffer a packet from a previous block that has not lost any packets so far and admits the arriving packet. Only if such an intact block cannot be found does the server delete a second packet from a block. Figure 3 depicts the algorithm by which packets are deleted from the buffer.

Suppose now that the server in question has a finite buffer that can store up to B packets, and that the arrival process consists of a stream of packets sent in ascending sequence numbers from a single source. Making the block size K less than B guarantees that whenever an arriving packet finds a full buffer, there is at least one packet from a previous block in the buffer. If $B \leq K$, arriving packets from the end of the i-th block may not find packets from the $(i-1)$-th block in the buffer, thereby limiting the usefulness of this technique. In the next section we show that although the best performance without buffer management is sometimes achieved for a block length larger than the buffer size, not much degradation is encountered when K is restricted to be smaller than or equal to B. This provides for further improvement by buffer management.

In the discussion so far we assumed that the packet stream arriving at the server consists of a consecutive sequence of contiguous packet blocks. Under these circumstances, the server has access at any moment to at most $\lceil B/K \rceil$ blocks aside from from the block whose packets are arriving at that moment. For a single packet recovery coding, $\lceil B/K \rceil + 1$ is the maximum length burst of missing packets that can be recovered.

If the level of spreading offered by the buffer management is not sufficient, the source can arrange its packets in N interleaved streams, by assigning to each stream every N-th packet. Such *deterministic interleaving*, which is similar to the array parity discussed in subsection 2.5, spreads bursts of deleted packets in the arriving stream over multiple blocks. However, monitoring multiplexed streams requires more effort by the server than is needed for a single stream.

"Natural" spreading of bursts occurs when the arriving stream comprises intermixed packets from many sources. This is the case, for example, when the server represents a controller of an output queue in a space-division switch [9], to which packets are arriving from all input ports, each carrying an independent stream. In this case the packet mix is random rather than deterministic, in the sense that a given packet belongs to a specific stream with some probability. It is interesting to note that deterministic interleaving offers better burst spreading and thus lower probability of loss than random interleaving. To see this, consider a burst of length $s \leq N$. All the packets in this burst are recoverable under deterministic interleaving because no block suffers more than one missing packet. On the other hand, in the random case, there is some probability that some blocks suffer two or more losses. This probability may be quite large, a phenomenon known as the "birthday paradox" [7], so named because of the "paradoxical" fact that among a group of 23 people, there is a better than 50% probability that two of them will share a birthday. In our case, a burst of 14 consecutive erasures in a data stream consisting of 100 randomly multiplexed streams results in 0.615 probability that two of the erasures will share the same traffic stream, thereby resulting in an unrecoverable erasure. In contrast, the use of a 100-way deterministically interleaved stream would guarantee that each of the erasures in the burst of 14 would occur in a different FEC block.

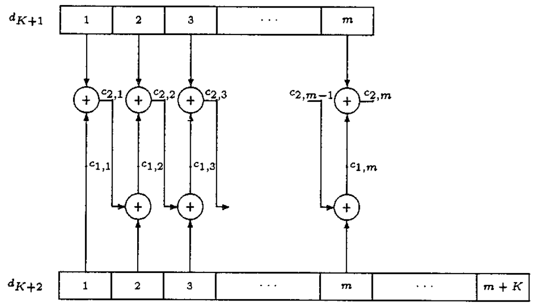

Figure 2: A decoder for reconstructing two packets

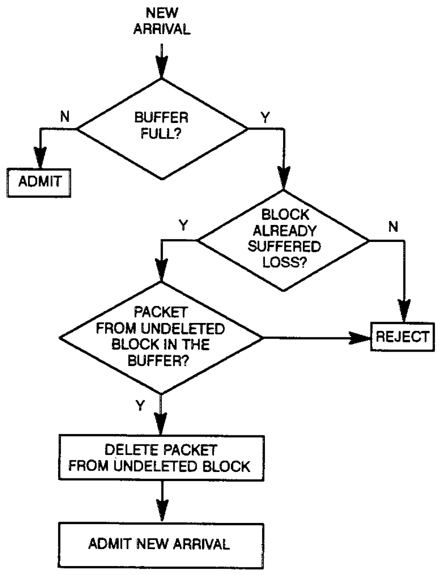
NEW ARRIVAL
BUFFER FULL?
N
Y
ADMIT
BLOCK ALREADY SUFFERED LOSS?
Y
N
PACKET FROM UNDELETED BLOCK IN THE BUFFER?
REJECT
Y
DELETE PACKET FROM UNDELETED BLOCK
ADMIT NEW ARRIVAL

Figure 3: An algorithm for packet rejection

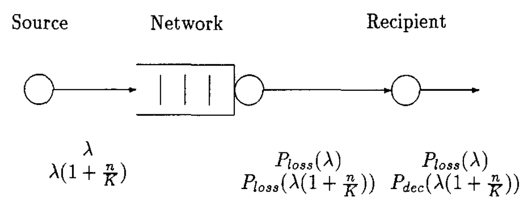

Figure 4: Performance evaluation model

4 PERFORMANCE EVALUATION

4.1 General considerations

As indicated above, the two factors in the packet loss process which must be incorporated in a model are:

- The packet rejection distribution and the effect of adding parity packets on that distribution
- The reduction in packet loss rate through buffer management and coding

The model we use in this section for ascertaining this effects is depicted in Figure 4. It consists of a data source to generate both data and parity packets and send them through a single server with a finite buffer, which represents the network. Packets are lost when an arriving packet finds a full buffer. The specific packet that is rejected in this case depends on the buffer management scheme exercised by the server. The data recipient gets the packet sequence and attempts to recover missing data packets by using the parity packets.

The main parameters in the model are the buffer size (B), the number of parity packets per block (n), the number of data packets in a block (K), and the rate, denoted by λ packets/second, at which the source generates fixed size (m bits) data packets. Time is slotted and each data packet requires exactly one slot to be transmitted. We assume that the number of packets arriving during slots $1, 2, \ldots$, are independent Poisson distributed random variables with mean λ. The measure of performance we use is the loss ratio, defined in the following paragraph.

When only data packets are sent, at rate λ, the loss rate $P_{loss}(\lambda)$ experienced at the server is also the loss rate observed at the recipient output assuming no retransmissions. Adding n parity packets to every block of K data packets, increases the packet rate at the server to $\lambda(1+\frac{n}{K})$ and consequently the loss rate of the packet stream to $P_{loss}(\lambda(1+\frac{n}{K}))$. This latter loss rate, however, is reduced by decoding at the recipient to $P_{dec}(\lambda(1+\frac{n}{K}))$. The loss ratio, G, of the coding scheme is defined as

$$G \triangleq \frac{P_{dec}(\lambda(1+\frac{n}{K}))}{P_{loss}(\lambda)} \quad , \qquad (5)$$

and for a coding scheme to be useful, it must have $G < 1$.

Several interesting tradeoffs can be investigated with this model. An example is the percent of parity bits in the packet stream. On the one hand, the number of packets that can be recovered in a block increases with the number of parity packets in the block. On the other hand, a large percent of overhead packets increases the traffic rate and hence the loss rate in the packet stream arriving at the decoder. The percent of overhead packets is also constrained by the requirement that the total packet rate, data and parity, should be less than 1 to avoid buffer saturation. We investigate these and other considerations in the subsections below with a simple analytic model and with more realistic simulation results. We end this section with a discussion of FEC code design constraints.

4.2 Analytic model

To analytically model the performance, we consider the case in which the packets arriving at the server are from a single source, and coding is based on adding a single parity packet to each block of K data packets with no interleaving. We assume that the numbers of packets arriving in time slots $1, 2, \ldots$ are independent, Poisson distributed random variables with rates λ and $\lambda(1+\frac{1}{K})$, for the uncoded and coded packet streams respectively.

The packet loss probabilities, $P_{loss}(\lambda)$ and $P_{loss}(\lambda(1+\frac{1}{K}))$ are each the rejection probability for a discrete-time, single-server queue with finite size B, and constant service time of one slot. The buffer behavior is modeled as a finite-state discrete time Markov chain, in which the state is the number of packets in the queue just before the beginning of a slot. The state transition probabilities can be found in the literature (for example, in reference [11]).

The decoder performance is evaluated under an additional assumption, which is that each packet finds a full buffer with probability $p = P_{loss}(\lambda(1+\frac{1}{K}))$, independently of other packets. That is, we represent the effect of buffer overflow by marking each packet with a "loss" tag with probability p and leave it unmarked with probability $1-p$.

Consider first decoding with no buffer management. At the decoder, the number of lost packets in a block is a random variable with binomial distribution and parameters $K+1$, p. That is,

$$\begin{aligned} Pr\{i \text{ lost packets in a block}\} &= b(K+1, i, p) \\ &\triangleq \binom{K+1}{i} p^i (1-p)^{K+1-i} \quad . \end{aligned}$$

A lost packet can be recovered if and only if it is the only lost packet in its block. The average number of packets lost in a block after decoding is given by

$$EL = \sum_{j=2}^{K+1} j b(K+1, j, p) = (K+1)p - (K+1)p(1-p)^K \quad . \quad (6)$$

The packet loss rate after decoding, P_{dec}, is thus given by

$$P_{dec} = EL/(K+1) = p - p(1-p)^K \approx p\left(1 - e^{-Kp}\right) \quad , \quad (7)$$

where the approximation is for small values of p and large values of K such that Kp is finite.

Let us now consider the selective rejection of packets at a typical network node. Suppose that the node buffer is not smaller than the block length, so that whenever an arriving packet, say from block l, finds a full buffer, there is at least one packet from block $l-1$. The new packet is accepted to the buffer only if a previous packet from block l has been rejected and block $l-1$ did not suffer any packet loss. In this case, a packet from block $l-1$ is rejected to make room for the new arrival from block l. This implies that block $l-1$ arrives with no lost packets if and only if block l has no more than one lost packet.

To ascertain the effect of this buffer management procedure on the decoder's performance, we start with the stream of rejected ("marked") and accepted packets. If two or more packets are missing from block l and none from $l-1$, the buffer management action amounts to deleting a packet from block $l-1$ and

	Loss Probability		
	before	after	with buffer
K	decoding	decoding	management
∞	$2.5488\ 10^{-5}$		
70	$4.1728\ 10^{-5}$	$1.2171\ 10^{-7}$	
19	$1.4972\ 10^{-4}$	$4.2528\ 10^{-7}$	$1.2794\ 10^{-8}$
10	$6.5533\ 10^{-4}$	$4.2805\ 10^{-6}$	$2.5084\ 10^{-7}$
8	$1.3547\ 10^{-3}$	$1.4601\ 10^{-5}$	$1.1187\ 10^{-6}$
6	$4.1127\ 10^{-3}$	$1.0024\ 10^{-4}$	$1.1517\ 10^{-5}$

Figure 5: Probability of packet loss with $B = 20$ and $\lambda = 0.8$; all quantities are for $\lambda(1 + \frac{1}{K})$

restoring a packet in block l. If a packet is already missing from block $l-1$, no "trading" is done.

Let i, j be the quantity of missing packets in blocks l and $l-1$, respectively. Let $L_{i,j}$ be the number of packets that cannot be recovered after decoding in block l, given i, j.

$$L_{i,j} = \begin{cases} 0 & i = 0, 1 \\ 0 & i = 2,\ j = 0 \\ i-1 & i \geq 3,\ j = 0 \\ i & i \geq 2,\ j > 0 \end{cases} \tag{8}$$

Thus, the probability of packet loss after decoding with buffer management is given by

$$\begin{aligned} P_{dec} &= \frac{EL}{K+1} \\ &= \frac{1}{K+1} \sum_{i=3}^{K+1} \{ib(K+1, i, p)[1 - b(K+1, 0, p)] \\ &\quad +(i-1)b(K+1, i, p)b(K+1, 0, p)\} \\ &= p - \frac{b(K+1, 1, p)}{K+1} + \frac{b(K+1, 0, p)}{K+1} \\ &\quad \cdot [1 - b(K+1, 1, p) - b(K+1, 0, p) + b(K+1, 2, p)] \end{aligned} \tag{9}$$

Figures 5 and 6 depict the performance of the scheme described above, for buffer size $B = 20$, and packet arrival rates before encoding of 0.8 and 0.9, respectively. The entries for $K = \infty$ represent no encoding, and the packet loss probabilities are those for a single server queue with the above arrival rates. The second row shows the packet loss rate for the value of K that minimizes that loss rate after decoding, without buffer management. For example, for $\lambda = 0.8$, with no encoding the packet loss rate is $2.55\ 10^{-5}$. With block size $K = 70$, the higher packet rate of $0.8(1 + 1/70)$ causes the loss probability to increase to $4.17\ 10^{-5}$. However, the decoder reduces that rate to $1.22\ 10^{-7}$, thereby achieving a total packet loss reduction of more than two orders of magnitude. For $K = 19$, the largest block size for which the server can apply the buffer management scheme effectively, the total improvement in P_{dec} is almost three orders of magnitude. At data-packet rate $\lambda = 0.9$, the initial loss rate is higher than before and the improvement is smaller, as Figure 6 shows. However, even at that congested level, the packet recovery scheme, along with proper buffer management, reduces the packet loss rate by more than an order of magnitude.

4.3 Simulation

In the analytic model we assume that packets are rejected independently. To study the sensitivity of the coding scheme to correlation in the rejection process, we simulated the model shown in Figure 4. Runtime parameters to the simulation include data input rate, the type of FEC, the amount of deterministic

	Loss Probability		
	before	after	with buffer
K	decoding	decoding	management
∞	$1.3547\ 10^{-3}$		
65	$2.1841\ 10^{-3}$	$2.8907\ 10^{-4}$	
19	$6.2846\ 10^{-3}$	$7.0736\ 10^{-4}$	$1.2221\ 10^{-4}$
10	$1.8904\ 10^{-2}$	$3.2561\ 10^{-3}$	$8.7815\ 10^{-4}$
8	$2.9961\ 10^{-2}$	$6.3854\ 10^{-3}$	$2.1102\ 10^{-3}$
6	$5.4272\ 10^{-2}$	$1.5084\ 10^{-2}$	$6.3900\ 10^{-3}$

Figure 6: Probability of packet loss with $B = 20$ and $\lambda = 0.9$

interleaving or random multiplexing, and the type of buffer management.

Each simulation run was repeated five times with different random number generator seeds. In all cases, the confidence intervals computed from these values are more narrow than the lines on the graphs. Although the effects of initial conditions were not eliminated, comparisons with analytic results, where available, indicate that the measured simulation results deviate by less than 10% from the computed values.

The following paragraphs compare the results of the simulation with those of the analytic model, then present simulation results comparing the efficacy of buffer management, FEC, deterministic interleaving, and random multiplexing. These results show that buffer management is necessary to achieve an acceptable loss ratio, that a simple single-erasure-correcting FEC scheme used in conjunction with deterministic interleaving is superior to more complex FEC schemes, that deterministic interleaving is superior to random multiplexing, and that loss ratios of 10^{-3} are achievable.

	Loss Probability		
	before	after	with buffer
K	decoding	decoding	management
∞	$2.68\ 10^{-5}$		
70	$4.79\ 10^{-5}$	$3.96\ 10^{-5}$	$3.45\ 10^{-5}$
19	$1.80\ 10^{-4}$	$1.53\ 10^{-4}$	$8.53\ 10^{-5}$
10	$6.67\ 10^{-4}$	$5.15\ 10^{-4}$	$1.98\ 10^{-4}$
8	$1.61\ 10^{-3}$	$1.21\ 10^{-3}$	$3.36\ 10^{-4}$
6	$3.85\ 10^{-3}$	$2.77\ 10^{-3}$	$5.61\ 10^{-4}$

Figure 7: Probability of packet loss with B = 20 and $\lambda = 0.8$ (simulation)

	Loss Probability		
	before	after	with buffer
K	decoding	decoding	management
∞	$1.61\ 10^{-3}$		
70	$1.87\ 10^{-3}$	$1.74\ 10^{-3}$	$1.66\ 10^{-3}$
19	$5.78\ 10^{-3}$	$5.14\ 10^{-3}$	$3.89\ 10^{-3}$
10	$1.47\ 10^{-2}$	$1.23\ 10^{-2}$	$6.69\ 10^{-3}$
8	$2.41\ 10^{-2}$	$1.95\ 10^{-2}$	$9.34\ 10^{-3}$
6	$4.68\ 10^{-2}$	$3.69\ 10^{-2}$	$1.45\ 10^{-2}$

Figure 8: Probability of packet loss with B = 20 and $\lambda = 0.9$ (simulation)

4.3.1 Comparison to Analytic Results Figures 7 and 8 show that the loss probability, both with and without buffer management, is much worse than that predicted by the analytic model; in fact, in all cases the FEC gain is insufficient to overcome the greater erasure rate caused by the addition of the redundant packets. This is due to the extremely bursty nature of the queue overflow process. The following sections evaluate some methods for improving this situation.

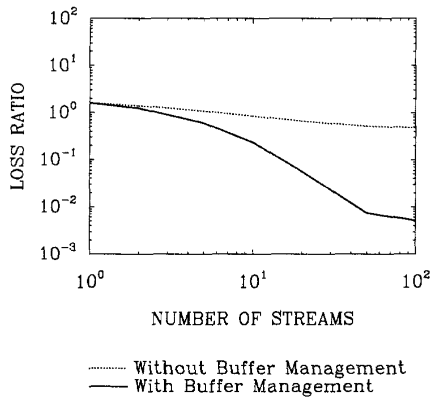

Figure 9: Loss ratio with and without buffer management

4.3.2 Buffer Management The great improvement due to buffer management is clearly illustrated in Figure 9, which shows the loss ratio with and without buffer management, respectively, for a M/D/1/10 queue carrying randomly multiplexed streams with an aggregate λ of 0.85. Each stream has single-erasure-correcting FEC, each block of which consists of 20 data packets and a single FEC packet. A loss ratio of zero indicates perfect performance.

As can be seen from the figure, buffer management improves the loss ratio by up to two orders of magnitude. Note that more than two traffic streams must be present in order for coding to show any benefit at all, even with buffer management. As will be shown later, deterministic interleaving may be used to overcome this difficulty.

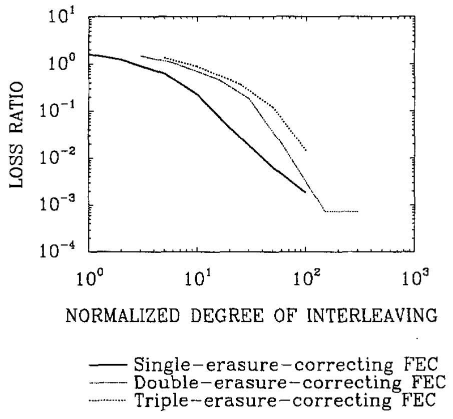

Figure 10: FEC versus deterministic interleaving

4.3.3 FEC Versus Deterministic Interleaving Another method of reducing the loss ratio is to increase the erasure-correcting capability of the FEC coding scheme, by using the diagonal parity packets described in previous sections. Recall that a diagonal parity packet is longer than a data packet. We make the pessimistic assumption that two m-bit packets are added to the traffic for each diagonal packet.

This section compares three FEC coding schemes: a single-erasure-correcting scheme requiring one overhead packet per block, a double-erasure-correcting scheme requiring three overhead packets per block, and a triple-erasure-correcting scheme requiring five overhead packets per block. These scheme are compared at equal code rates; thus the first scheme has 20 data packets per block, the second has 60, and the third has 100.

The single-erasure-correcting scheme with a three-way interleave can be considered to act as a separate coding scheme with 60 data packets per block that is capable of correcting from one to three erasures per block, depending on the distribution of erasures within the block. Similarly, the single-erasure-correcting scheme with a five-way interleave can be considered to act as a separate coding scheme with 100 data packets per block that is capable of correcting from one to five erasures per block, again depending on the distribution of erasures within the block.

Figure 10 demonstrates that interleaving the single-erasure-correcting scheme is superior to using the more complex multiple-erasure-correcting schemes for a M/D/1/10 queue with buffer management and λ = 0.85. The loss ratio for each FEC coding scheme is plotted against a normalized degree of interleaving consisting of the actual degree of interleaving multiplied by the ratio of the block length divided by the block length of the single-erasure-correcting scheme. Figure 10 thus compares the double-erasure-correcting scheme to the single-erasure-correcting scheme at triple the interleave, and compares the triple-erasure-correcting scheme to the single-erasure-correcting scheme at quintuple the interleave.

Note the presence of loss ratios of better than 10^{-3} in Figure 10.

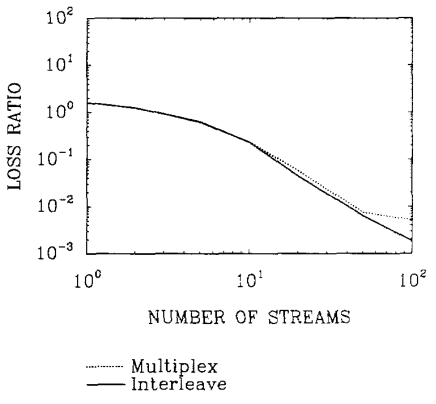

Figure 11: Interleaving versus multiplexing

4.3.4 Interleaving versus multiplexing Figure 11 shows that deterministic interleaving is slightly superior to random multiplexing for an M/D/1/10 queue with buffer management and load λ = 0.85. This effect becomes more pronounced for larger queues, which have lower raw loss rates.

However, the increased interleaving results in increased delay for correction of erased packets. The tradeoff between delay and reliability is application-dependent. It is instructive to compare the delay caused by interleaving with the delay imposed by round-trip time on retransmissions. Assuming that the source and destination are located 3000 km from each other, communicate at a rate of 1 gigabit/second, and are connected by a speed-of-light communications medium, the round-trip time will be greater than 1600 packet durations. This means that an FEC scheme that uses a block length of 20 packets may use up to 160-way interleaving while still maintaining an average reconstitution delay smaller than is possible with a retransmission-based scheme.

5 CONCLUSION

In this paper, we have presented a novel technique for reducing packet loss rate in high speed wide area networks. Parity packets, which the source adds to each block of data packets, are used by the recipient to reconstruct lost packets from the respective blocks. The missing packets are identified by observing a sequence-number gap in the stream of incoming packets. Methods for recovering a single, double, and larger numbers of losses from a block were presented. Bit errors can be corrected by either reconstructing the affected packet as if it were missing or by additional parity information.

To alleviate the effect of packet loss correlation, we proposed to enhance buffer management procedures in the networks so that packets are rejected based on their block affiliation and the number of packets already lost in their block. A similar effect can be achieved by interleaving the data, either intentionally by the source, or as it occurs naturally during statistical multiplexing.

We evaluated the performance of these packet recovery schemes using a model, which consists of a source that codes the data; a single-server, discrete-time, finite-capacity queue which causes packet loss and at which buffer management is carried out; and a recipient, which uses a decoder to reconstruct missing packets. The performance measure we selected was the ratio of packet loss rate after decoding to the rate when no coding is used. This ratio, denoted as loss ratio, must be smaller than 1 for the code to be useful. Both analytic and simulation results were presented for this model. The former, obtained under the assumption of independent packet losses, showed loss ratios of the order 10^{-3} and smaller.

The simulation runs showed that packet loss correlation is a severe problem when each packet stream has a dedicated finite buffer. In this case buffer management and/or interleaving is vital to the achievement of good loss ratios. We have also shown that deterministic interleaving achieves better loss ratios than statistical multiplexing. This observation implies that when multiple streams are intermixed, as in output queues of a space-division switch, round-robin polling is superior to random polling. Loss ratios of better than 10^{-3} were demonstrated.

In summary, our performance study shows that significant reduction in packet loss rate can be achieve with a combination of coding, buffer management, and interleaving.

References

[1] A.S. Accampora. An overview of lightwave packet networks. *IEEE Network*, 3(1):29–41, January 1989.

[2] R.E. Blahut. *Theory and Practice of Error Control Codes.* Addison-Wesley, 1958.

[3] H. Burton and D. Sullivan. Error and error control. *Proceedings of the IEEE*, 60(11):1293–1301, Nov 1972.

[4] S.N. Chiou and V.O.K. Li. An optimal two-copy routing scheme in a communication network. In *Proceedings of INFOCOM'88*, New Orleans LA,, 1988.

[5] G. Clark and J.B. Cain. *Error-Correcting Coding for Digital Communications.* Plenum Press, 1981.

[6] IEEE 802.6 Committee. Proposed standard: Distributed queue dual bus (DQDB) Metropolitan area network, August 1989.

[7] W. Feller. *An Introduction to Probability Theory and its Applications.* John Wiley, 1958.

[8] P. Gonet, P. Adam, and J.P. Coudreuse. Asynchronous time division switching: The way to flexible broadband communications networks. In *Proc. Int. Zurich Sem. Digital Commun.*, pages 141–145, Zurich, Switzerland, March 1986.

[9] M. Hluchyj and M. Karol. Queueing in high-performance packet switching. *IEEE Transactions on Communications*, COM-36(12):1587–1597, December 1988.

[10] A. Huang and S. Knauer. Starlite: a wideband digital switch. In *Proc. of Globecom'84*, pages 121–125, 1984.

[11] M. Karol, M. Hluchyj, and S. Morgan. Input versus output queueing on a space-division packet switch. *IEEE Transactions on Communications*, COM-35(12):1347–1356, December 1987.

[12] N.F. Maxemchuck. The manhattan street network. In *Proc. of Globecom'86*, pages 255–261, New Orleans LA,, 1986.

[13] E. Nussbaum. Communication network need and technologies–a place for photonic switching? *IEEE Journal on Selected Areas in Communications*, 6(7):1036–1043, August 1988.

[14] P.R. Prucnal, D.J. Blumenthal, and M.A. Santoro. 12.5 gbit/s fiber-optic network using all-optical processing. *Electronics Letters*, 23(12):629–630, June 1987.

[15] R. Rom and N. Shacham. Reconfiguration algorithm for a double-loop token passing ring. *IEEE Transactions on Computers*, 37(2):182–189, February 1988.

[16] Y. Sun and M. Gerla. SFPS: a synchronous fast packet switching architecture for very high speeds. In *Proc. of INFOCOM'89*, pages 641–646, Ottawa, Canada, 1989.

[17] J. Turner. New directions in Communications. *IEEE Communication Magazine*, 24(10), 1986.