Performance Evaluation of Forward Error Correction in an ATM Environment

Ernst W. Biersack, *Member, IEEE*

***Abstract*— If the packet loss rate in a network is higher than the loss rate requested by an application, the transport protocol must make up for the difference in loss rate. In high bandwidth delay–product networks, the latency introduced by retransmission-based error recovery schemes may be too high for applications with latency constraints. In this case, *forward error correction* (FEC) can be used. FEC allows recovery from loss without retransmission. The amount of loss recovered strongly depends on the loss behavior of the network. FEC works best if losses are dispersed in time.**

We use simulation to study the loss behavior of an output buffered cell multiplexer for three different traffic scenarios. Our results show how the loss behavior of the multiplexer is affected by the traffic mix and the statistics of the sources. The more bursty the sources, the higher the loss rate and the higher the probability that losses will occur in bursts. Based on simulation results, we develop a mathematical model for the performance of FEC, when applied to multiplexed traffic, and compute the effectiveness of FEC for the three traffic scenarios. FEC is not effective for the two homogeneous traffic scenarios. However, FEC reduces the loss rate for the video sources by several orders of magnitude for a heterogeneous traffic scenario consisting of video and burst sources.

I. Introduction

THE asynchronous transfer mode (ATM) is the internationally accepted transfer mode for broad-band networks [2]. ATM provides high-bandwidth, low-latency multiplexing and switching [15]. The basic unit of multiplexing and switching in ATM is called a *cell*. A cell has a fixed length of 53 bytes: 48 bytes of data (*payload*) and 5 bytes of control information such as *virtual path identifier* (VPI), *virtual channel identifier* (VCI), and a *cyclic redundancy check* (CRC) for header error control. One of the reasons for adopting ATM is for the integration of services. ATM will be used by different applications that require services with widely varying *quality of service* (QOS) requirements. One QOS requirement is reliability of the transfer. An ATM-type network will experience three types of errors: bit errors corrupting the data portion of a cell, switching errors due to undetected corruption of the cell header, and cell losses due to congestion. Losses due to congestion are expected to be far more common than the other two types of errors.

If the reliability provided by the network is lower than the reliability requested by an application, the error control system in the end nodes must make up for the difference. The two basic mechanisms available to improve reliability are *automatic repeat request* (ARQ) and *forward error correction* (FEC).

Automatic repeat request (ARQ) is a *closed-loop* technique based on retransmission of data that were not correctly received by the receiver. ARQ requires the transmitter and receiver to exchange state information about the status of individual messages [20]. Each retransmission of a message adds at least one round-trip time of latency. Therefore, ARQ may not be applicable for transmitting data from applications with low latency constraints. Low latency is necessary for applications such as human interaction (voice, video), process control, remote sensing, etc. If data do not arrive within a certain time, they will be worthless for these applications. Another disadvantage of ARQ-based schemes is the processing overhead required to keep track of a potentially very large number of outstanding messages.

Found error correction (FEC) is an alternative to ARQ. It avoids the shortcomings of ARQ, and is well suited for operation in high bandwidth–delay product networks, where the ratio of the packet transmission time to the propagation delay is large. FEC involves the transmission of redundant information along with the original data so that if some of the original data is lost, it can be reconstructed using the redundant information. The amount of redundant information is typically small, so that FEC remains efficient. In the past, FEC has been used in computer memories [10] compact disks [23], communication with deep space probes [4], and video coding [11]. In data communications, the use of FEC is attractive for providing reliability as needed without increasing the end-to-end latency. FEC can make the operation of the network more cost-effective by allowing it to operate with higher utilization. Without FEC, the network must operate at a utilization level where the loss rate of the network never exceeds the most stringent loss rate required by any application. In this case, all applications will receive this low loss rate, independent of their actual need.

When FEC is used, there are two antagonistic effects at work:

1) the redundant information due to FEC helps recover part of the losses

2) the additional data due to FEC increase the overall load, which makes the loss rate worse.

FEC is only effective if 1) prevails. The potential of FEC to recover from losses depends very much on the loss behavior of the network. If the losses are highly correlated, FEC will be far less effective than when the losses are dispersed evenly. For

Manuscript received February 11, 1992; revised July 23, 1992.

The author was with Bell Communications Research, Morristown, NJ 07960. He is now with EURECOM, Sophia Antipolis, 06560 Valbonne, France.

IEEE Log Number 9206811.

this reason, we study the loss behavior of a cell multiplexer for different traffic scenarios before we evaluate the performance of FEC.

In the next section, we explain the operation of FEC. Section III contains a description of our simulation model and the assumptions made. For different traffic scenarios, we study in Section IV the loss behavior of a multiplexer, and evaluate in Section V the performance of FEC. Related work is discussed in Section VI, and the results are summarized in Section VII. The Appendix contains more results about the loss behavior of the multiplexer and an algorithm for computing the performance of FEC.

II. Operation of Forward Error Correction

Coding theory distinguishes two types of data corruption: an *error* is defined as a bit with an unknown value in an unknown location, whereas an *erasure* is a bit with an unknown value in a known location. If the FEC decoder is able to take advantage of erasure information, replacing an error with an erasure approximately doubles the error correcting power of the code. In ATM, 1 byte of the 5 byte header is a cyclic redundancy check for header error control that can correct 1 b errors and detect 2 b errors. For bit error rates smaller than 10^{-9}, which are typical for fiber optics transmission systems, the cell loss rate due to random bit errors is negligible. Therefore, congestion losses are the dominant form of error on ATM networks, and the network can be modeled as a well-behaved erasure channel. The unit of loss is a cell. Congestion losses may occur in bursts and affect consecutive cells of a single connection. For FEC to be effective, it is important to recover the loss of consecutive cells. Cells must contain a sequence number to enable the receiver to detect missing cells that must be reconstructed.

The FEC system we study in this paper uses a Reed–Solomon based *burst erasure* correcting code, referred to as RSE [14]. RSE takes k data cells as input and produces h redundant cells as output. The k data and h redundant cells are transmitted to the receiver. Any k of the $k+h$ transmitted cells are sufficient to recover the original k cells as long as none of the received cells is corrupted by bit errors (erasures only). A group of k cells from which the h redundant cells are generated is referred to as a *block*. The h redundant cells are referred to as *overcode*. A block is the unit of error detection and loss recovery. A block that has at least one cell missing is said to be *corrupted*. A block of k data cells and h redundant cells is considered *lost* if more than h of the $k+h$ cells are missing. FEC is applied by individual sources. A source that applies FEC is referred to as a *FEC source;* a source that does not apply FEC is a *non-FEC source*. Non-FEC sources transmit *data blocks* that consist of k data cells. FEC sources transmit *FEC blocks* that consist of k data and h redundant cells.

Fig. 1 illustrates the operation of the RSE system for $k = 3$ and $h = 2$ cells.[1] The RSE encoder at the transmitter produces two redundant cells (FEC 1, FEC 2) for every block of three data cells. In the example, one data cell (MSG 2) and one redundant cell (FEC 1) get lost. The FEC decoder at the receiver uses the redundant cell FEC 2 and the two correctly received data cells to reconstruct the missing data cell (MSG 2). RSE can be implemented on a single chip, and achieves a throughput of 400 Mb/s or 1 Gb/s, depending on the symbol size used. Since RSE can only correct erasures but not errors, its decoding algorithm is simplified and the same hardware can be used for encoding and decoding, which reduces the overall complexity. Thanks to the dramatic increase in chip integration and speed, an implementation of RSE that runs at B–ISDN rates is feasible. One of the advantages of RSE over other Reed–Solomon codes [18] is that the original data cells are not modified and can be transmitted in clear form immediately, without waiting for the encoding process to complete. This also implies that there is no decoding delay at the receiver if all k data cells are received. In general, the encoding and decoding delays for the RSE chip are negligible compared to the transmission delay. When operating at 400 MHz on 53 byte cells with a symbol size of $m = 8$ b on blocks of $k = 50$ data and $h = 10$ redundant cells, the encoding delay d is $d = 8.005\,\mu$s. The encoding delay is defined as the time between the availability of k data cells at the encoder and generation of the first redundant cell. When operating at 1 GHz with $m = 32$ b, the encoding delay for RSE is reduced to $d = 1\,\mu$s. At the receiver, decoding is only necessary if fewer than k data cells are received. The decoding must be deferred by a waiting time w until k error-free cells (data and redundant cells) of an FEC block are received. The time w depends on the rate at which cells of an FEC block arrive and on the number of cells missing. Since ATM does not reorder cells, we have $w <$ (*average cell interarrival time*) $* h$. If we choose $k = 50$ [cells], $h = 10$ [cells], and assume an average cell interarrival time of $60\,\mu$s, we get $w < 60 * 10\,\mu$s. After the time w has elapsed, the decoding of the missing cells can start. The time to decode the missing cells is the same as the encoding delay d. Therefore, the total delay for encoding and decoding introduced by FEC is $d + (w + d)$. For the above example, the total delay will be less than $616.010\,\mu$s, which is at least an order of magnitude smaller than the transmission delay introduced in high bandwidth–delay product networks.

The buffer requirements when FEC is used are the same as without FEC. If the receiver performs a reassembly of cells into larger units, it must provide a reassembly buffer that is large enough to hold one block of cells. If the receiver

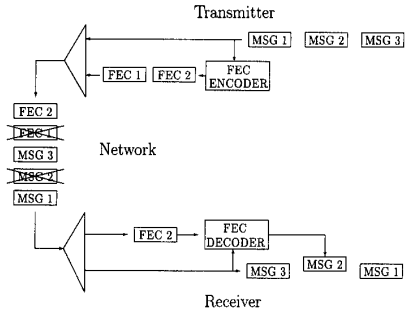

Fig. 1. Operation of FEC.

[1] Typically, the ratio k/h is much larger than in this example.

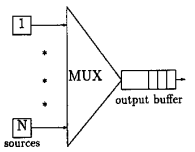

Fig. 2. Model of the output-buffered multiplexer.

operates in a "cut-through" manner and each cell contains sufficient header information to determine its position within an application data unit, an incoming cell can be put right into application memory without the need for intermediate buffering [6].

Our measure for the performance of FEC will be the effect on the block loss rate. Since higher layers deal with blocks, the block loss rate is a more meaningful measure of the network performance than the cell loss rate. FEC increases the total amount of traffic in the network by the number of redundant cells generated by the FEC encoder. If λ is the aggregated load generated by all sources, the *effective load* λ_{eff} in the network consists of the aggregate load λ and the load due to FEC. If all sources apply FEC, $\lambda_{\text{eff}} = \lambda(1 + h/k).\lambda$ and λ_{eff} assume values between 0 and 1, where a load of 1 means that cells are generated at the same rate as the system can transmit them.

III. Experimental Setup

We have built a discrete event simulator to study the loss behavior of an ATM network and to investigate the performance of FEC. We model the ATM network as a multiplexer. Since simulations are very CPU-intensive, it is currently impossible to simulate a more detailed ATM environment consisting of, for instance, several switches connected by high-speed links. The multiplexer has N input ports and one output port, as shown in Fig. 2. Cells are buffered at the output port in a single shared buffer of finite capacity of B cells. The multiplexer checks the inputs in a round-robin fashion, and puts newly arriving cells in the output buffer, if there is any space, and drops them otherwise. Each input port has one source connected to it. Every source generates, on the average, the same amount of traffic. The cells waiting in the output buffer are served in FIFO order. The granularity of time in the simulation is a *cell time,* which is equal to the service time of a single cell.

An ATM network will carry traffic from different types of applications with different statistics. In our simulation model, we distinguish between two different types of sources: 1) *burst sources* representing applications such as bulk data transfer or transactions, and 2) *video sources* representing variable bit rate video sources such as entertainment video or video conferences. A burst source is characterized by the *interarrival time among bursts, burst size,* and *separation* of cells within each burst. We assume that the burst interarrival time is geometrically distributed, the number of cells per burst is a constant, and the separation of cells within a burst is fixed. A separation of x means that during the transmission of a burst, a cell is transmitted every x cell times. For the burst sources, different values of λ are obtained by changing the average burst interarrival time.

TABLE I
Simulation Parameters

Parameter	Value
Load λ	0.3 – 0.99
Total number of sources N	32
Number of video sources *v_srcs*	0, 24, 32
Number of burst sources *b_srcs*	0, 8, 32
Burst interarrival time	geometric distribution
Burst size	50 [cells]
Block length k	50 [cells]
Cell separation for video sources	equidistant over frametime
Cell separation *sep* for burst sources	10 [cells]
Size B of output buffer	100 [cells]

For video traffic, there are many encoding schemes (e.g., MPEG, JPEG, H.261, DVI [1], but very few acceptable statistical models for variable bit rate video [8], [13], [9]. A shortcoming of all these models is that they do not capture the long-term correlation of variable bit rate video traffic [3]. We therefore do not use a statistical model to generate the traffic for the video sources. Instead, the arrival statistics of our video sources are derived from entertainment video. The data set for the video sources has been generated by encoding a 2 h-long action movie using an intraframe 8×8 discrete cosine transform coding scheme with run-length and Huffman encoding [7]. The data set contains the number of bytes per frame produced by the encoder. The duration of one frame is $\frac{1}{24}$ s; the total number of frames is 171,000. The statistics for the video data are

- maximum bit rate: 15.06 Mb/s
- mean bit rate: 5.34 Mb/s
- minimum bit rate: 1.79 Mb/s
- maximum /mean: 2.82.

In the simulator, the data for each frame are fragmented into cells, and the transmission of the cells is spaced out equally over the duration of the frame. The different video sources are *unsynchronized,* i.e., they start at different points in the movie. Unsynchronized video sources represent a situation where different people watch the same movie at different times (video on demand). When FEC is applied to a video source, the spacing between cells is adjusted such that the original cells together with the redundant cells are spaced equidistantly over the duration of one frame. Since the video sources have a fixed bandwidth, the capacity of the multiplexer is altered to yield different values for the load λ. The simulation parameters are listed in Table I.

We use the following three scenarios.

- Scenario V32 has 32 video sources that are all unsynchronized, buffer size 100, and block length of 50.
- Scenario B32 has 32 burst sources, block length 50, burst length 50, cell separation 10, and buffer size of 100.

• Scenario VB24-8 has 24 video sources and 8 burst sources, block length 50, and a buffer of 100. The burst sources have burst length 50 and cell separation 10.

We have also performed simulations with other values for the parameters listed above. In particular, we varied the output buffer size, block length, and burst size. While a change in any of these parameters affects the cell and block loss rates,[2] we found that it did not affect the cumulative distribution function $F_{CL}(x)$ of the percentage of cells lost in a corrupted block that determines the effectiveness of FEC and will be defined below. Considering the results obtained from all our simulations, we believe that the results presented in the following are representative for the three scenarios.

IV. Cell Loss of an ATM Multiplexer

The block loss rate and the cell loss pattern of a multiplexer greatly affect the effectiveness of FEC. The block loss rate $P_{\text{Data}}(\lambda)$ is defined as $P_{\text{Data}}(\lambda) \stackrel{\text{def}}{=} \Pr$ *(At least one cell is lost in a data block* | *load* = λ), with $P_{\text{Data}}(\lambda) \in [0,1]$. FEC is only effective if the FEC decoder recovers enough lost cells to reduce the block loss rate after decoding to a level lower than the block loss rate without FEC.

• Since the loss rate increases with load, FEC will increase the block loss rate before decoding. As the transmission of redundant cells increases the load from λ to λ_{eff}, the block loss rate will increase from $P_{\text{Data}}(\lambda)$ to $P_{\text{Data}}(\lambda_{\text{eff}})$.

• Since FEC can recover only up to h missing cells in an FEC block of $k+h$ cells, it is particularly effective when cell losses are spread uniformly and corrupted blocks have few cells lost. FEC is not effective when bursts of cells are lost and the corrupted blocks have a large number of cells lost. The higher the percentage of corrupted blocks that have fewer than h cells lost, the more corrupted blocks can be reconstructed, i.e., the more effective is FEC.

In the following, we study the block loss rate and the loss behavior of the multiplexer for the three different scenarios. Fig. 3 shows the block loss rate as a function of the load λ. For B32, losses start to occur at loads as low as $\lambda = 0.3$. For V32, no losses were observed for $\lambda < 0.9$, and for $\lambda \geq 0.9$, the losses increase steeply. As we will see later, this steep increase will have a negative impact on the effectiveness of FEC for scenario V32. The block loss rate of the burst sources and the video sources in scenario V24-8 are almost identical, which indicates that more than the statistics of the source itself, it is the traffic mix entering the multiplexer that determines the block loss rate.

The burst sources are on–off sources, and a source is either idle or transmits at a rate of $\frac{1}{10}$ of the capacity of the multiplexer. The video sources continuously generate cells at a rate that depends on the encoded frame.[3] The arrival pattern for the video sources changes no more than every $\frac{1}{24}$ s, which is several orders less frequent than in the case of burst sources. Fig. 4 depicts the distribution of the multiplexer buffer filling

[2] An increase in the buffer size reduces the loss rates; an increase in the burst size increases the loss rates.

[3] The mean bit rates for the video sources and the burst sources are approximately the same.

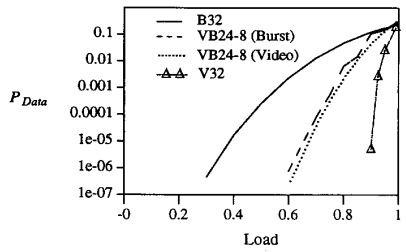

Fig. 3. Block loss rate P_{Data} for the different scenarios (no FEC is used).

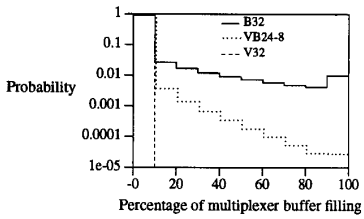

Fig. 4. Multiplexer buffer filling for the different scenarios, $\lambda = 0.7$ (no FEC is used).

for the different scenarios. We see that for B32, the probability that the multiplexer buffer is between 90% and 100% full is more than two orders of magnitude higher than for VB24-8. On the other hand, for V32, the buffer is never loaded more than 10%.

To capture the cell loss behavior, we measure the number of cells lost in a data block. Let CL be a discrete random variable that represents the percentage of cells lost in a *corrupted* data block, i.e., a block with at least one lost cell. $CL \in [1,100]$. $F_{CL}(x) \stackrel{\text{def}}{=} \Pr(CL \leq x)$ is the cumulative distribution function of the percentage of cells lost in a corrupted block. $F_{CL}(x) = p$ means that with probability p, the percentage of lost cells in a corrupted block is $\leq x$. The larger the value of $F_{CL}(x)$ for a given x, the more corrupted blocks have $\leq x$ percent of the cells lost. Therefore, the larger $F_{CL}(x)$, the better FEC will perform because the more corrupted blocks can be reconstructed. We also consider the percentiles of CL. The α percentile of CL, $0 \leq \alpha \leq 100$ is defined as the smallest integer x_α for which $F_{CL}(x_\alpha) \geq \alpha/100$, i.e., ${}^{+}d \stackrel{\text{def}}{=} \min\{x \mid F_{CL}(x) \geq \alpha/100\}$. The percentiles of CL are indicators for the effectiveness of FEC. The lower the value of a percentile, the more corrupted blocks can be reconstructed by the FEC decoder with a certain amount of overcode.

A necessary condition for the effectiveness of FEC is that most of the corrupted blocks can be recovered, i.e., have fewer cells lost than the FEC decoder can reconstruct. In Fig. 5, we see $F_{CL}(x)$ for the three different scenarios for $\lambda = 0.7$ and $\lambda = 0.9$. For any value of x, $F_{CL}(x)$ is highest for V32 and lowest for B32, which implies that, on average, a corrupted block loses more cells for scenario B32 than for scenario V32. For scenario VB24-8, the values of $F_{CL}(x)$ are higher for the video sources than for the burst sources and lie between the ones for scenario V32 and scenario B32. These results

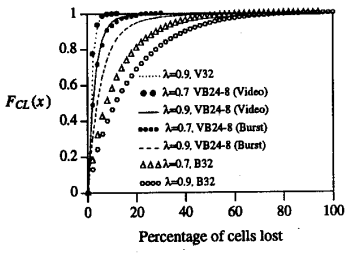

Fig. 5. $F_{CL}(x)$ for the different scenarios (no FEC is used).

TABLE II
PERCENTILES OF CL FOR ALL SCENARIOS, $\lambda = 0.9$ (NO FEC IS USED)

Scenario	90.0	99.0	99.9
	Percentiles of CL		
V32	4	6	6
B32	40	72	86
VB24-8, Video	8	18	26
VB24-8, Burst	14	34	54

indicate that the value of $F_{CL}(x)$ depends on the behavior of a source—the smoother a source, the higher the value of $F_{CL}(x)$—and on the traffic mix between the video sources and the burst sources. Note also that for each scenario, as the load increases, the value for $F_{CL}(x)$ decreases for a fixed x. This means that for increasing load, more overcode is necessary to recover a certain percentage of corrupted blocks. More details on the loss behavior are contained in [5].

In Table II, we magnify the region of $F_{CL}(x)$ with $F_{CL}(x) \geq 0.90$ and give the 90.0, 99.0, and 99.9 percentiles for $\lambda = 0.9$. In the case of a video source, 99% of all corrupted blocks[4] lose $\leq 6\%$ of their cells for V32 and $\leq 18\%$ of their cells for VB24-8. For a burst source, the percentage of cells lost by 99% of the corrupted blocks is higher, with up to 34% for VB24-8 and up to 72% for B32. Therefore, orders of magnitude more corrupted blocks can be recovered if a certain amount of overcode is applied by a video source as compared to a burst source.

Section A of the Appendix contains further observations about the loss behavior of the multiplexer.

V. PERFORMANCE OF FEC

A. Performance Measures

The transmission of redundant data increases the load from λ to λ_{eff} and affects the block loss rate of both FEC and non-FEC sources. The data block loss rate for a non-FEC source increases from $P_{\text{Data}}(\lambda)$ to $P_{\text{Data}}(\lambda_{\text{eff}})$. For FEC sources, the FEC-block loss rate $P_{\text{FEC}}(\lambda_{\text{eff}})$ measures the block loss rate after the FEC decoder tried to reconstruct the missing cells of corrupted blocks. The FEC-block loss rate is defined as $P_{\text{FEC}}(\lambda_{\text{eff}}) \stackrel{\text{def}}{=} \Pr(\textit{At least } h+1 \textit{ cells are lost in an FEC block} \mid \text{load} = \lambda_{\text{eff}})$. The reconstruction of an FEC block with $k+h$ cells will fail if more than h cells are missing.

[4]If the FEC decoder can recover 99% of all corrupted blocks, the block loss rate will be reduced by two orders of magnitude.

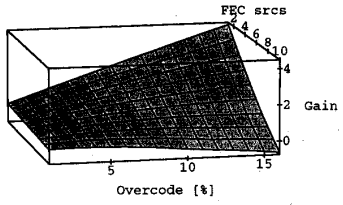

Fig. 6. Gain for scenario V32, $\lambda = 0.9$.

To evaluate the performance of FEC, we define three performance measures. The *gain* G for an FEC source is defined as $G \stackrel{\text{def}}{=} \log_{10}(P_{\text{Data}}(\lambda)/P_{\text{FEC}}(\lambda_{\text{eff}}))$. G measures the reduction of the block loss rate in terms of orders of magnitude. When FEC is used by a subset of the sources, the non-FEC sources will always observe an increase in their block loss rate. The *penalty* D for a non-FEC source expresses the relative increase of the block loss rate. It is defined as $D \stackrel{\text{def}}{=} \log_{10}[P_{\text{Data}}(\lambda_{\text{eff}})/P_{\text{Data}}(\lambda)]$. Gain and penalty measure the effect of FEC on an individual source. The *aggregate gain* AG combines gain and penalty and expresses the effectiveness of FEC from a systems point of view. The aggregate gain is defined as $AG \stackrel{\text{def}}{=}$ (*number of FEC sources*) $*$ *gain* $-$ (*number of non-FEC sources*) $*$ *penalty*. The aggregate gain measures the net orders of block loss rate reduction due to FEC. FEC is effective if $AG > 0$.

For all three scenarios, we will plot three-dimensional graphs that show the gain, penalty, and aggregate gain as a function of overcode and number of FEC sources. The results displayed in these graphs are obtained from a mathematical model for the performance of FEC. The motivation for this model and its derivation is given in the Appendix. We checked the validity of the model by additional simulations.

B. Performance of FEB for Scenario V32

We first consider scenario V32, where all sources are video sources. Fig. 6 shows the gain at load $\lambda = 0.9$. For V32, most corrupted blocks lose only very few cells (see Table II). Also, the increase in load and in block loss rate is marginal when only one source applies FEC. Therefore, if FEC is applied by a single source, the gain is significant and increases steadily with the amount of overcode. As the number of FEC sources grows, the increase in load due to FEC causes a significant increase of the block loss rate and completely cancels out the reduction of the block loss rate due to the reconstruction of corrupted blocks. As a consequence, the gain becomes zero or even negative. The sensitivity of the block loss rate to small increases in the load is reflected in the high values for the penalty, which is shown in Fig. 7. Fig. 8 shows the aggregate gain. The aggregate gain is never positive, which means that the increase in block loss rate for the non-FEC sources is higher than the reduction in block loss rate for the FEC sources.

Overall, FEC is not effective for scenario V32. Instead of using FEC to achieve a certain block loss rate, the aggregate

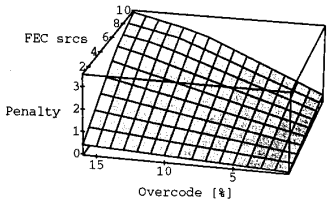

Fig. 7. Penalty for scenario V32, $\lambda = 0.9$.

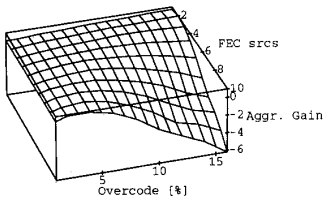

Fig. 8. Aggregate gain for scenario V32, $\lambda = 0.9$.

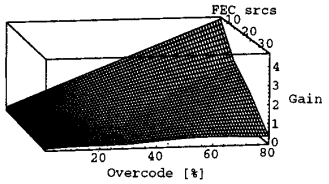

Fig. 9. Gain for scenario B32, $\lambda = 0.4$.

load λ must be reduced to meet the most stringent block loss rate of any source.

C. Performance of FEC for Scenario B32

For scenario B32, we study the effectiveness of FEC at load $\lambda = 0.4$ and load $\lambda = 0.7$. If only part of the sources applies FEC, their gain is positive (see Figs. 9 and 12). For instance , if a single source applies FEC with an overcode of 40%, the gain at $\lambda = 0.4$ is $G = 2.3$ and at $\lambda = 0.7$ is $G = 1.65$. The lower gain for $\lambda = 0.7$ is due to the fact that the average number of lost cells per corrupted block increases with increasing load. As the number of FEC sources increases, the gain decreases to zero. This is due to the fact that the block loss rate increases and cancels out the reduction in block loss rate by the FEC decoder and makes. The penalty for B32, see Figs. 10 and 13, stays in about the same range as the gain for B32. The aggregate gain for $\lambda = 0.4$, as shown in Fig. 11, is positive with values between 0 and 12 and increases as the number of FEC sources or the amount of overcode increases. While the aggregate gain is highest if a large number of sources applies a very high overcode, the gain for an individual source in this case is marginal ($G < 1$). The aggregate gain for $\lambda = 0.7$ (see Fig. 14) is never positive. For combinations where either the overcode or the number of FEC sources is smaller than 10, the aggregate gain is about zero. As the amount of overcode or the number of FEC sources increases, the effective load λ_{eff} approaches 1 and the aggregate gain grows negative.

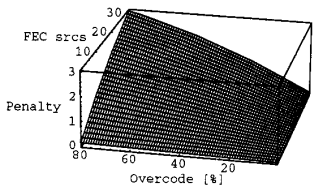

Fig. 10. Penalty for scenario B32, $\lambda = 0.4$.

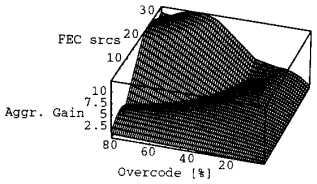

Fig. 11. Aggregate gain for scenario B32, $\lambda = 0.4$.

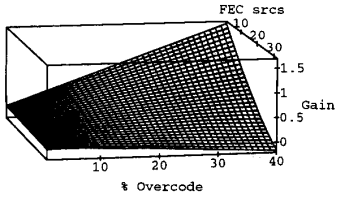

Fig. 12. Gain for scenario B32, $\lambda = 0.7$.

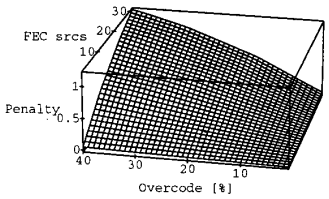

Fig. 13. Penalty for scenario B32, $\lambda = 0.7$.

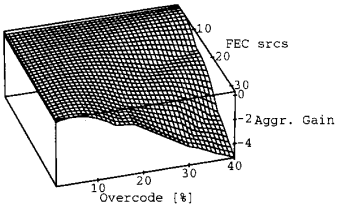

Fig. 14. Aggregate gain for scenario B32, $\lambda = 0.7$.

In general, FEC is not effective for scenario B32. The only situation where the gain is noticeable is at load $\lambda = 0.4$ if a few sources apply a high amount of overcode. Therefore, ARQ should be used if burst sources represent applications that can tolerate the extra latency due to retransmissions. The ARQ-based loss recovery scheme may retransmit only the messages that are lost and make the most efficient use of the bandwidth.

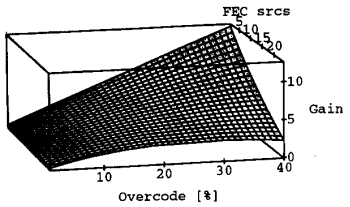

Fig. 15. Gain for video sources in scenario VB24-8, $\lambda = 0.7$.

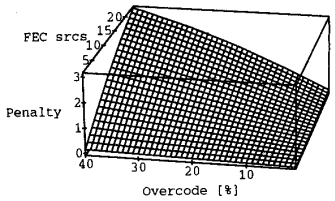

Fig. 16. Penalty for video sources in scenario VB24-8, $\lambda = 0.7$.

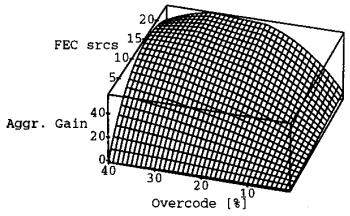

Fig. 17. Aggregate gain for scenario VB24-8, $\lambda = 0.7$.

D. Performance of FEC for Video Sources in Scenario VB24-8

The last scenario considered is VB24-8 with 24 video sources and 8 burst sources. Because of space limitations, we consider only the case where video sources apply FEC. Fig. 15 shows the gain for the video sources. For a small number of FEC sources, the gain increases linearly with the overcode and can become higher than 10. As the number of FEC sources approaches the total number of video sources, increasing the amount of overcode has less effect on the gain. Only up to an overcode of about 20% does the gain increase. For overcodes higher than 20%, the block loss rate grows at about the same rate as the capability of FEC to recover corrupted blocks. Therefore, increasing the amount of overcode beyond 20% does not result in any further increase of the gain. Fig. 16 shows the penalty for the video sources, which assumes values between 0 and 3. The penalty for the burst sources, which is not shown here, is about the same.

For VB24-8, the aggregate gain is always positive, see Fig. 17, and its absolute values indicate that FEC is very effective. We also studied configurations that are slightly different from the one presented here. We could observe a high aggregate gain for other loads also and for other ratios between the number of video sources and the number of burst sources. As the load λ increases or the ratio between the number of video sources and the number of burst sources decreases, the aggregate gain of FEC decreases.

TABLE III
COMPARISON OF THE LOSS BEHAVIOR FOR THE THREE SCENARIOS

Values for	Scenario V32	B32	VB24-8
$\frac{\partial P_{Data}}{\partial \text{load}}$	high	medium	medium
Percentiles of CL	low	very high	medium

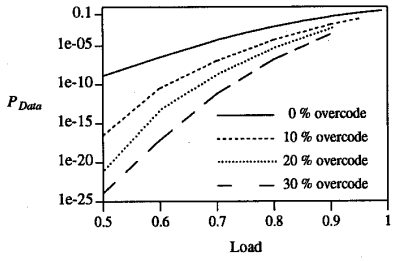

Fig. 18. Block loss rate P_{Data} for different degrees of overcode for scenario VB24-8 with eight video sources applying FEC.

The fact that FEC is effective only for scenario VB24-8 can be explained by the results that were contained in Fig. 3 and Table II. We summarize the results in Table III.

We see the following.

- FEC is not effective for V32 because of the high value for $\partial P_{\text{Data}}/\partial_{\text{load}}$. The increase in load due to FEC results in a significant increase of the block loss rate P_{Data}.
- FEC is not effective for B32 because of the very high values for the percentiles of CL. To recover a significant portion of the corrupted blocks, a high amount of overcode is necessary. The increase in load due to the use of FEC results in a penalty for the non-FEC sources that makes FEC ineffective.
- FEC is effective for VB24-8 because of the moderate values for the percentiles of CL and for $\partial P_{\text{Data}}/\partial_{\text{load}}$. A small amount of overcode reduces the loss rate for the FEC sources significantly and affects the loss rate of the non-FEC sources only marginally.

Fig. 18 plots for scenario VB24-8 the block loss rate for the video sources applying FEC. The load and the degree of overcode vary; the number of video sources applying FEC is fixed at eight. The higher the amount of overcode, the higher is the reduction of the block loss rate. The block loss rate reduction is highest for low loads. This is due to the fact that the values of $F_{CL}(x)$ are decreasing with increasing load, see Fig. 5. As the load approaches $\lambda = 1$, the increase in load due to FEC almost entirely cancels the loss reduction due to FEC.

Although maximizing the aggregate gain is attractive, it may not always be possible because of other constraints that limit the feasible combinations of overcode and number of FEC sources. Such constraints are, for instance, upper bounds on the block loss rates of the FEC sources and the non-FEC sources. To compute the aggregate gain plotted in Fig. 19, two such constraints are introduced: 1) the block loss rate for the FEC sources must be less than 10^{-6}, and 2) the block loss rate for the non-FEC sources must be less than 10^{-3}. The original block loss rate without using FEC is $5.1 * 10^{-5}$.

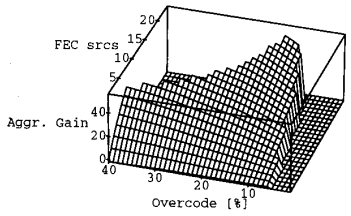

Fig. 19. Modified aggregate gain for scenario VB24-8, $\lambda = 0.7$.

All combinations of overcode and number of FEC sources that violate at least one of these constraints have been assigned the aggregate gain of zero. We see in Fig. 19 that there are two regions that violate either one of the constraints. Small values for the overcode are not sufficient to yield the required block loss rate for the FEC sources. As the overcode increases, the number of FEC sources must be reduced to meet the required block loss rate for the non-FEC sources. This result suggests that the number of FEC sources should be controlled to limit the penalty for the non-FEC sources. The number of sources with a block loss rate of less than 10^{-6} is maximal for an overcode of 14%.

VI. Related Work

Loss recovery using FEC has been studied previously [19], [17], [24]. Our work distinguishes itself in several ways.

- Our traffic model for the video sources is derived from a real movie, while previous models used to evaluate the performance of FEC assume that the interarrival times are exponentially or hyperexponentially distributed.
- A modified Reed–Solomon burst erasure correcting code (RSE) is used that leaves the data cells unmodified and can correct about twice as many erasures (lost cells) as a standard Reed–Solomon code. The RSE code is able to recover any h cells lost out of $k + h$ cells.
- We study the effect of FEC on both the FEC sources and the non-FEC sources, while most of the other papers focus on the performance for the FEC sources only.

Shacham and McKenny study the performance of FEC based on the ex–OR operation to generate one or two redundant cells in a block of k cells [19]. They conclude that a mechanism that recovers at most one or two lost cells in a block is not very effective because cells can get lost in bursts. To alleviate the impact of consecutive losses and to improve the performance of their FEC scheme, they suggest deterministic interleaving of cells from different streams or intelligent dropping of cells when the buffer overflows. Interleaving, however, increases the end-to-end delay.

Researchers from NTT Japan study the effectiveness of a FEC scheme based on the ex–OR [17]. To improve the effectiveness of their scheme, a sequence of cells is arranged as a two-dimensional array (matrix) with the ex–OR operation performed over each row and column. One redundant cell is generated per row and column. If the number of columns is h, the scheme can recover up to h consecutive cell losses.[5]

[5] The redundant cells generated over the rows are used for loss *detection*.

A simple model is developed to calculate the reduction of the cell loss rate due to FEC. The results indicate that a significant cell loss reduction of up to ten orders of magnitude can be achieved. However, their model uses small p-values $p \in [0.01, 0.4]$.

Zhang [24] devises a Markov model to describe the cell loss behavior for a switching node with a finite buffer and multiple service priorities and computes the cell and block loss rate. Reed–Solomon codes are used to generate the redundant cells. The cell loss rate reductions achievable can be as high as 10^{15}. However, the model does not consider the load increase due to FEC and assumes that the cell arrivals follow a Poisson distribution.

VII. Conclusion

We have studied the loss behavior of a cell multiplexer and the performance of FEC for three different traffic scenarios. The loss behavior depends on the statistics of the source and on the traffic scenario. From simulations, we observed that the percentage of cells lost in a block is geometrically distributed. Based on that observation, we can compute $F_{CL}(x)$, the cumulative distribution function of the percentage of cells lost per block. In a hybrid approach, the results obtained via simulation are used to compute the gain, penalty, and aggregate gain of FEC.

FEC trades bandwidth for latency to improve the loss rate. FEC should be used to support constrained latency applications such as video sources that can tolerate some loss. For FEC to be effective, it is necessary that most corrupted blocks lose only a few cells. The gain for the FEC sources varies with the load, the amount of overcode, and the relative number of sources using FEC. For a fixed overcode, as the number of FEC sources increases, the gain decreases and the penalty increases. The aggregate gain is high for a heterogeneous traffic scenario with FEC applied by the video sources. For the two homogeneous traffic scenarios, the aggregate gain is small or negative. Our results indicate that FEC is very effective for the heterogeneous traffic scenario.

Our results obtained for the cell multiplexer demonstrate that the FEC is a viable way to achieve the loss performance required by video sources while operating the multiplexer at high utilization. FEC has the advantage that it is solely performed in the end system. For video, other techniques have been suggested to achieved good picture quality when transmitting video data over packet-switched networks. One such technique is layered coding [22] in combination with selective cell dropping by the switches [12]. The encoded data are separated in layers of different importance.[6] The high-frequency components of an image contribute less to the overall quality of the picture. Cells carrying only high-frequency components can be dropped in case of switch overload. Such a scheme requires that each cell has a priority field that allows a switch to discriminate cells. By discarding low-priority cells, it is hoped that the high-priority cells carrying the essential information can get transmitted without

[6] Layered encoding as compared to nonlayered encoding increases the amount of data injected into the network.

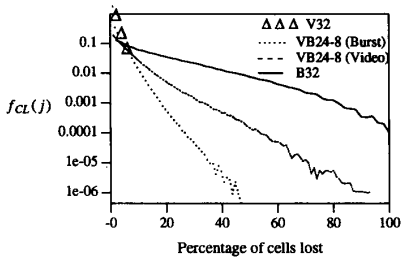

Fig. 20. Probability density function $f_{CL}(j)$ of the cells lost in a corrupted block for the different scenarios, $\lambda = 0.9$ (no FEC is used).

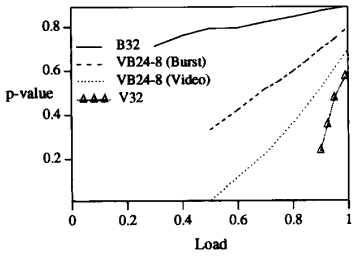

Fig. 21. p-values for the different scenarios (no FEC is used).

loss. Layered coding relies on the support of priorities through the switches. FEC, on the other hand, does not make any assumption about particular features provided by the switches, and is therefore more generally applicable. Considering the proliferation of different switch architectures [21], [16], we should not expect that in the future every switch will support priorities. It is, however, conceivable that FEC is used in connection with layered encoding to protect the high-priority data that must not be lost, in case not every switch along the transmission path supports priorities.

APPENDIX

A. Loss Behavior of Multiplexer

Fig. 20 shows the probability density function $f_{CL}(j)$ for the percentage of cells lost in corrupted block.[7] The smaller the gradient of a curve in Fig. 20, the higher the probability that a block loses more than one cell.

When plotted on a linear/logarithmic scale, all the curves are approximately straight lines.[8] Therefore, the random variable CL is geometrically distributed with $f_{CL}(j) = (1-p) * p^{j-1}$. For a particular scenario and load, the cell loss behavior for corrupted blocks is completely described by a single parameter p, which will be referred to as the *p-value*. We will use this observation in Section B to develop a mathematical model for the effectiveness of FEC. $f_{CL}(1)$ can be obtained from simulation, and we can compute the p-value as $p = 1-f_{CL}(1)$.

The p-value expresses the probability that in a corrupted block, more than one cell is lost. The lower the p-value, the higher the probability that a corrupted block can be recovered with a certain amount of overcode. Fig. 21 shows the p-values for the different scenarios and source types as a function of the load. The p-values are highest for scenario B32. For the other scenarios, the p-values are significantly smaller and change more as the load increases.

B. Calculation of the Performance Measures

We use a hybrid approach to obtain quantitative results for the three performance measures, gain, penalty, and aggregate gain. The observation that $f_{CL}(j)$ can be approximated by the geometric function allows us to compute $F_{CL}(x)$ and drastically reduces the number of simulation runs needed. First, simulations are performed, where no FEC is applied, to measure the block loss rates and the p-values. Second, the block loss rates and the p-values are used to compute the performance of FEC for an arbitrary combination of number of FEC sources and degree of overcode. To compute for any particular scenario the gain, penalty, and aggregate gain of FEC as a function of the overcode, the number of FEC sources, and the load, the following steps are executed.

1) Define a scenario by fixing the parameters: number of video and burst sources, buffer size, burst length, block length, and cell separation.

2) Perform a set of simulations for different loads $\lambda_1, \cdots, \lambda_M$ to obtain a set $\mathcal{B}$ of block loss probabilities $\mathcal{B} = \{P_B(\lambda_1), \cdots, P_B(\lambda_M)\}$ and a set of p-values $\mathcal{P} = \{p_{\lambda_1}, \cdots, p_{\lambda_M}\}$.

3) Compute the gain, penalty, and aggregate gain for FEC:

a) Fix the load λ, the percentage of overcode *ocode*, and the number of FEC sources f_srcs.

b) Compute $\lambda_{\text{eff}} = \lambda + (\lambda * ocode/100 * f\_srcs)$.

c) Use $\mathcal{B}$ and $\mathcal{P}$ to calculate via interpolation $P_B(\lambda_{\text{eff}})$ and $p_{\lambda_{\text{eff}}}$.

d) Compute $F_{CL}(ocode) = \sum_{j=1}^{ocode} (1 - p_{\lambda_{\text{eff}}}) * p_{\lambda_{\text{eff}}}^{j-1}$ and $P_{\text{FEC}}(\lambda_{\text{eff}}) = P_{\text{Data}}(\lambda_{\text{eff}}) * (1 - F_{CL}(ocode))$.

e) Compute the gain $G = \log_{10}(P_{\text{Data}}(\lambda)/P_{\text{FEC}}(\lambda_{\text{eff}}))$, the penalty $D = \log_{10}(P_{\text{Data}}(\lambda_{\text{eff}})P_{\text{Data}}(\lambda))$, and the aggregate gain $AG = f\_srcs * G - (N - f\_srcs) * D$.

ACKNOWLEDGMENT

I would like to thank M. W. Garrett for providing the data for the video source and C. J. Cotton for writing an early version of the simulator. The comments of S. J. Golestani, A. J. McAuley, Lj. Trajković, and C. J. Cotton on earlier versions of this paper are very much appreciated. I would also like to thank the anonymous reviewers for their constructive comments.

REFERENCES

[1] P. H. Ang, P. Ruetz, and D. Auld, "Video compression makes big gains," *IEEE Spectrum*, pp. 16–19, Oct. 1991.

[2] Comité Consultatif International de Télégraphique et Téléphonique, "Broadband aspects of ISDN," Recommendation I.121, 1989.

[3] J. Beran, R. Sherman, M. S. Taqqu, and W. Willinger, "Variable-bit-rate video traffic and long-range dependence," Submitted for publication, 1992.

[4] E. R. Berlekamp, R. E. Peile, and S. P. Pope, "The application of error control in communications," *IEEE Commun. Mag.*, vol. 25, pp.44–57, Apr. 1987.

[7] Given $f_{CL}(j)$, $F_{CL}(x)$ is defined as $F_{CL}(x) \stackrel{\text{def}}{=} \sum_{j=1}^{x} f_{CL}(j)$.

[8] The same observation was made in [17] for other source models while measuring the number of consecutive cells lost.

[5] E. W. Biersack, "A simulation study of forward error correction in ATM networks," *Comput. Commun. Rev.*, vol. 22, pp. 36–47, Jan. 1992.
[6] E. W. Biersack, C. J. Cotton, D. C. Feldmeier, A. J. McAuley, and W. D. Sincoskie, "Gigabit networking research at Bellcore," *IEEE Network Mag.*, vol. 6, pp. 42–49, Mar. 1992.
[7] M. W. Garrett and M. Vetterli, "Congestion control strategies for packet video," in *Proc. 4th Int. Workshop Packet Video,* Kyoto, Japan, Aug. 1991.
[8] R. Grünenfelder *et al.*, "Characterization of video codes as autoregressive moving average processes and related queueing systems performance," *IEEE J. Select. Areas Commun.*, vol. 9, pp. 284–293, Apr. 1991.
[9] S. S. Huang, "Modeling and analysis for packet video," in *Proc. GLOBECOM '89,* Dallas, TX, Nov. 1989, pp. 881–885.
[10] S. Kaneda and E. Fujiwara, "Single bit error correcting double bit error detecting codes for memory systems," *IEEE Trans. Comput.*, vol. C-31, pp. 596–602, July 1982.
[11] K. Kawashima and H. Saito, "Teletraffic issues in ATM networks," *Comput. Networks ISDN Syst.*, vol. 20, pp. 369–375, 1990.
[12] H. Kröner, "Comparative performance study of space priority mechanisms for ATM networks," in *Proc. INFOCOM '90,* San Francisco, CA, June 1990, pp. 1136–1143.
[13] D. Maglaris *et al.*, "Performance models of statistical multiplexing in packet video communications," *IEEE Trans. Commun.*, vol. 36, pp. 834–844, July 1988.
[14] A. J. McAuley, "Reliable broadband communications using a burst erasure correcting code," in *Proc. ACM SIGCOMM '90,* Philadelphia, PA, Sept. 1990, pp. 287–306.
[15] S. E. Minzer, "Broadband ISDN and asynchronous transfer mode (ATM)," *IEEE Commun. Mag.*, vol. 27, pp. 17–24, Sept. 1989.
[16] P. Newman, "ATM technology for corporate networks," *IEEE Commun. Mag.*, vol. 30, pp. 90–101, Apr. 1992.
[17] H. Ohta and T. Kitami, "A cell loss recovery method using FEC in ATM networks," *IEEE J. Select. Areas Commun.*, vol. 9, pp. 1471–1483, Dec. 1991.
[18] M. O. Rabin, "Efficient dispersal of information for security, load balancing, and fault tolerance," *J. ACM,* vol. 36, pp. 335–348, Apr. 1989.
[19] N. Shacham and P. McKenny, "Packet recovery in high-speed networks using coding," in *Proc. INFOCOM '90,* San Francisco, CA, June 1990, pp. 124–131.
[20] J. D. Spragins, J. L. Hammond, and K. Pawlikowski, *Telecommunications: Protocols and Design.* Reading, MA: Addison-Wesley, 1991.
[21] F. Tobagi, "Fast packet switched architectures for broadband integrated services digital networks," *Proc. IEEE,* vol. 78, pp. 133–167, Jan. 1990.
[22] M. Vetterli, "Multi-dimensional subband coding: Some theory and algorithms," *Signal Processing,* vol. 6, Apr. 1984.
[23] L. B. Vries and K. Odaka, "CIRC—The error correcting code for the compact disk," presented at *the AES Premier Conf.* —The New World of Digital Audio, June 1982.
[24] L. Zhang and K. W. Sarkies, "Modelling of a virtual path and its applications for forward error recovery coding schemes in ATM networks," in *Proc. SICON '91,* Singapore, Sept. 1991.

Ernst W. Biersack (M'89) received the M.S. and Ph.D. degrees in computer science from the Technische Universität at München, Munich, Germany.

For the academic year 1982–1983 he was awarded a Fellowship from the Graduate School of the University of North Carolina at Chapel Hill. From 1983 to 1989 he was a Research Scientist at the Technische Universität München . From March 1989 to February 1992 he was a Member of the Technical Staff with the Computer Communications Research District, Bell Communications Research, where he was working on the design of a new transport protocol. Since March 1992 he has been an Associate Professor in Telecommunications at Institut EURECOM, Sophia Antipolis, France. His research interests include protocols for high-speed networks, internetworking, and performance evaluation of protocols.

Dr. Biersack is a member of the ACM.