Using forward error correction to protect real-time data against congestion errors on IP networks

Conference Paper · September 2004

CITATIONS
0

READS
218

2 authors, including:

Hendrik Christoffel Ferreira
University of Johannesburg
377 PUBLICATIONS **4,184** CITATIONS

SEE PROFILE

Using Forward Error Correction to Protect Real-Time Data Against Packet Loss

D.J.J. Versfeld
Department of Electrical and Electronic Engineering
Rand Afrikaans University
P.O. Box 524, Auckland Park, 2006
South Africa
Email: jv@ing.rau.ac.za

H.C. Ferreira
Department of Electrical and Electronic Engineering
Rand Afrikaans University
P.O. Box 524, Auckland Park, 2006
South Africa
Email: hcf@ing.rau.ac.za

A.S.J. Helberg
School for Electrical and Electronic Engineering
Potchefstroom University
for Christian Higher Education
P.O. Box 20862, Potchefstroom, 2522
South Africa
Email eeiasjh@puknet.puk.ac.za

***Abstract*— Packet loss on normal packet-switched networks occurs when the finite length buffer components comprising the network starts to overflow due to congestion [1]. On optical networks, a major source of packet loss can be contributed to contention errors [2]. Packet loss can degrade the quality of service (QoS) of real-time data substantially. One method that can be used to complement current techniques to combat packet loss on normal networks is to use forward error correction. Here, any network component that can be modelled as a finite queue with service rate μ_r can become a potential source of packet loss. An important factor influencing the QoS of real-time data when using forward error correction is the time in which the lost data can be recovered, as data that is delayed too long can be regarded as lost. In this paper the authors give an overview of existing decoders and techniques (especially a modified interleaving technique) found in literature and design a decoder based on the combination of two decoding techniques. The authors then compare the different decoders to determine which decoder has the optimum performance. The authors also propose a possible implementation of forward error correction in a normal packet-switched network to alleviate the effect of congestion errors and obtain results by simulation of a finite length queue with forward error correction.**

I. Introduction

The Quality of Service (QoS) on packet-switched networks are largely determined by the amount of packets lost, the delay experienced and the jitter experienced. The main source of packet loss on normal networks is buffer overflow due to congestion [1], hence packet losses on these networks can also be regarded as congestion errors. On optical networks, bursts can be dropped due to contention [2].

The conventional approach to combat packet loss on normal networks is to differentiate between the different classes of traffic and then to apply buffer management(FIFO, RED) and/or scheduling (WFQ, CB-WFQ or LLQ) and specialized routing/switching (MPLS, Intserv, Diffserv).

Alternative schemes to compliment the schemes above to alleviate the effect of packet-loss on real-time data exist. According to Perkins et al. [3], these techniques can be divided into sender-driven and receiver-driven techniques. Receiver-driven techniques use signal processing techniques to conceal losses. Sender-driven techniques can be divided into two main classes: retransmission techniques and Forward Error Correction (FEC) techniques. The advantage of FEC techniques is that a minimal delay is introduced as opposed to retransmission techniques. This minimal delay comes, however, at the expense of a higher bandwidth necessary for retransmission.

An important feature of normal packet-switched networks is that error detection is done at the lower layers of these networks, discarding erroneous packets when detected. Furthermore, the packets are numbered. Thus, with packet-switched networks, the receiver either receives error-free packets, or packets do not arrive at all [4]. On optical networks, the bit-error probability is very low, around 10^{-6}. These factors allow that packet losses can be modelled as erasures, reducing the complexity of the FEC schemes significantly.

In the survey, Perkins et al. [3] go further and divides FEC techniques into two classes: media-independent techniques and media specific techniques. The media specific techniques at the application layer include methods developed by Bolot et al. [5] and Hardman et al. [6]. Media independent techniques include methods developed by McAuley [7], Xu et al. [8], Berlekamp [9], Kamali et al. [10], Arai et al. [11], Blömer et al. [1], Luby et al. [12] and [13]and Rizzo [4]. In this paper, the authors focus on linear media-independent FEC codes to recover from lost packets.

The rest of the paper is organized as follows. In Section II, we will introduce a network protocol that uses forward error correction to alleviate the effect of packet loss on normal networks. In Section III, an overview is given of existing erasure decoders found in literature. In Section IV, the authors develop a combined decoder that proves to be quite efficient under certain circumstances. In Section V, the various decoders are compared for various scenarios. Finally, in Section VI, results will be given on a simulation where FEC was used to recover packets dropped from a finite length queue with with a constant bit-rate service time.

II. A FEC-enabled network protocol for Real-time data

When Forward Error Correction is used to combat congestion errors, extra packets containing recovery information is computed and transmitted with the original data packets.

Usually the FEC encoder calculates $n-k$ recovery packets from k data packets and transmits all n packets over the network. From any k packets received, the decoder will be able to reconstruct the original messages.

There are two important factors influencing the effectiveness of the FEC scheme. These factors are the following:

- the packet loss process of the network,
- the delay of the packets introduced by the FEC scheme.

We will first consider issues regarding the packet loss process and then issues regarding the delay.

Cidon et al. [14] analyzed the packet loss processes of high-speed networks. One of the major findings in the paper regarding FEC is that packet losses tend to occur in bursts at a finite-length buffer queue, such as a router or a switch. The paper also confirms that FEC packets increases the overall data rate on the network, which leads to a higher packet loss probability. The paper suggests that for a fixed-size block, continuous-time single session system, forward error correction will not perform too good. For fixed-size block, continuous-time multiple systems, it was found that the correlation between lost packets of the same session decreases as the rate of the session decreases for a given arrival rate to the system.

From the results of Cidon et al. [14], it is clear that not all packets can be protected by the FEC scheme. Thus, one possibility is to protect only the high priority interactive real-time data such as VoIP and Video. Furthermore, the total bandwidth used for the recovery packets should make out a very small percentage of the total bandwidth.

Another result from the analysis of Cidon et al. is that the lower rate data streams have a lower loss probability. As the loss process is quite bursty, the probability of consecutive packets being lost is quite high. Thus, if we lower the rate of the data stream we wish to protect, the probability of consecutive losses will also decrease for the stream in question.

From Cidon et al. it is clear that the FEC recovery packets can also become a source of congestion, leading to packet loss. This leads to three important characteristics that a FEC scheme should have. The scheme should utilize codes that add the minimum amount of redundancy which can recover the packet losses, i.e., maximum distance separable codes are preferable. The scheme should be rate adaptive in order to cater for the current loss process. The scheme should also deploy a back off scheme when there are too many packet losses, otherwise the FEC data will become a major contributor of packet loss, and due to the adaptive scheme, the whole process will have a snowball effect.

The second important factor of the FEC scheme is the delay introduced. Real-time packets have specific time slots in which they should be played out. If a packet is delayed beyond the time slot, the packet can be regarded as lost. Thus, encoding and decoding time of the FEC scheme needs to be minimized in order to achieve a high QoS.

The delays introduced by the forward error correction scheme can be divided into two categories:

- encoder(/decoder) wait delay, and
- processing delay.

Encoder(/decoder) wait delay is defined as the time the encoder(/decoder) has to wait for data before the data can be processed. This delay usually depends on the network architecture, i.e., the layer where the FEC-scheme is implemented, as well as whether the codes used to implement the scheme is systematic or non-systematic.

The FEC scheme can be implemented at the application layer, see for instance [5] and [6] or at the lower layers [7]. When implementing the scheme at the network layer, the encoder and decoder wait delays can be drastically reduced.

When considering an FEC scheme at the application layer, it is noticed that packets from the same source only can be grouped together. For the worst case scenario, we assume that packet one, two and three were grouped together and a redundant packet was calculated. If packet one is lost, while the other three packets are received, the decoder has to wait for the last three packets before packet one can be recovered. Assuming that each data packet contains 40ms of speech, the total delay before decoding can start is about 120ms. When we implement the scheme at the network layer, different users' data can be grouped together. If a packet is lost, we have to wait a maximum of 40ms (assuming each packet contains 40ms of speech) before decoding can start. This scheme differs also from the scheme proposed by McAuley [7] in that only a portion of the data is protected by FEC, i.e., the real-time data is protected, while the normal data relies on existing techniques like ARQ.

Another factor influencing the decoder wait delay is whether the code is a systematic or non-systematic code. For non-systematic codes, the decoder have to wait for k out of n packets, before any data can be released. With systematic codes, only a copy of the data is needed. The original packet can be forwarded as soon as a copy is made. With this scheme only the recovered packets will incur delays, while the received packets will experience no significant delays. For the encoder, we also have to wait for k packets before encoding can start, thus delaying the data packets as well. Another disadvantage of non-systematic codes is that when the code's erasure correcting capability is exceeded, i.e., more packets were lost than can be recovered, no data can be released at all, while with systematic codes, the received data packets can still be forwarded to their destinations.

Processing delay can be divided into encoder processing delay and decoder processing delay. We will be mainly interested in the decoder processing delay. The decoder processing delay is defined as the time the decoder takes to decode the data once all the packets are received. Decoding delay can be significantly reduced for the erasures-only case. However, various decoding strategies for the same code can differ greatly as will be discussed in Section V.

Using the above we will know describe a FEC protocol to alleviate the effect of packet loss on normal packet-switched networks. In Section VI we will give some results on the performance of such a scheme. The FEC scheme will be

designed in order to minimize the delay of recovered packets, while introducing no significant delay to the original data packets. The FEC scheme will also take into consideration the loss process of the packet-switched network.

A typical scenario is depicted in Fig. 1. We assume that we have x sources transmitting real-time interactive data, each transmitting at a rate of 1 packet per 40ms. We also assume that each user uses the same codec, such that each packet has the same size. In addition to the real-time data, the network also transports normal data, where the bandwidth of the normal data is much higher than that of the real-time data sources combined. From each time interval consisting of 40ms, a transmission group is formed of the x received packets, and additional recovery packets are calculated and transmitted with the real-time data packets at the FEC encoder agent. All the traffic then enters a finite-length buffer queue, which drops packets when the buffer starts to overflow due to overloading. The FEC decoding agent receives the transmission group. If data packets were lost and enough recovery packets was received, the FEC decoding agent decodes the dropped packets.

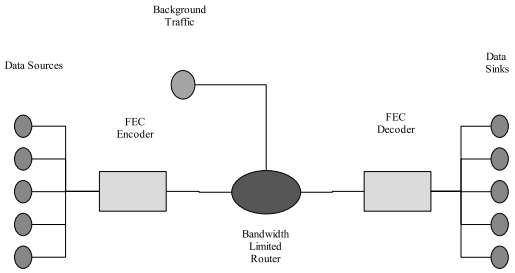


Fig. 1. The setup of the FEC scheme

The FEC encoding agent appends a FEC protocol header to every received packet. This is to identify each transmission group, and each member in the transmission group. The FEC encoding agent takes the first k incoming packets and group them together. The data as well as the IP and UDP headers of each packet is used to calculate the recovery packets. This simplifies the process at the FEC decoding agent. When data packets were lost, the IP and UDP headers can also be recovered during the decoding process.

The FEC decoder agent can record the channel state using some parameters as indication (the jitter, delay or number of packets lost). These data can then be conveyed to the FEC encoding agent via a feedback path. Based upon these statistics, the FEC encoding agent can adapt the amount of redundancy needed, or the size of the transmission group. A properly designed adaptive algorithm would also include a back-off algorithm.

In the next section, we will give an overview of some erasure codes that can be found in literature.

III. Overview of current FEC techniques

Various approaches to decode erasures using systematic codes were developed or investigated by among others, Berlekamp [9], McAuley [7], Kamali et al. [10], Xu et al. [8] and Rizzo [4].

Berlekamp introduced a technique that can be used to recover erasures from any linear binary code $C(x)$ by computing a vector space transformation for the code $C(x)$ based on the erasure positions. McAuley developed an erasure only algorithm for Reed-Solomon codes to decode erasures based on solving simultaneous equations where the erased symbols form the variables of the equations. The equations are formed by substituting the roots of the generator polynomial into the received codeword, knowing that the result of the substitution should be zero for a valid codeword. With this process, up to h independent equations can be formed that can solve up to h erased symbols, given that $h = n - k$. Techniques like Gaussian elimination can be used to solve the equations. (Gaussian elimination needs order N^3 operations.) Xu et al. have a very similar approach as McAuley, but they exploit the fact that the parity-check matrix of a Reed-Solomon code is a Vandermonde matrix with optimized inversion. Using an inverted submatrix of the parity-check matrix, together with the syndrome, they recover the erasures. Kamali et al. observed that the most expensive procedures of Reed-Solomon decoding based on the Berlekamp-Massey algorithm are the calculation of the error locator polynomial and the calculation of its roots, which can be omitted when dealing with erasures only. Thus, only the Forney algorithm is needed for the erasures-only case when using Reed-Solomon codes, greatly reducing decoding time.

Rizzo [4] used Vandermonde codes to protect data against packet loss. The Vandermonde codes are non-systematic linear block codes based on Lagrange's interpolation. Vandermonde codes can be transformed into systematic codes.

Both approaches from Xu and Rizzo exploits the fact that Vandermonde matrix inversion, which forms the basis of the erasure decoding process, can be done using N^2 operations. However, when the Vandermonde codes are transformed into systematic form, the matrix inversion can only be achieved using normal techniques, which uses N^3 operations.

A modified interleaving scheme can simplify the problem significantly, see [4], [10] and [1]. With interleaving and systematic encoding, only a copy of the data is needed, while the data can be forwarded to the next segment of the network. At the decoder, again only a copy of the data is needed, while the data can be forwarded to the next stage. Thus, the unaffected data experiences no real delay. Only lost packets will experience delay when recovered using the FEC decoder.

If the interleaving scheme is designed carefully, it can expedite the decoding process significantly. Luckily, this is not a difficult task. Refer to Fig. 2. The packets are written vertically into the columns of an interleaving matrix. The redundancy are computed using the data in the horizontal rows. The rate of the scheme can be adjusted by adjusting the rate of the

code and using the corresponding data and redundant columns. Each cell in the interleaver can contain more than one symbol. The key of this design is to duplicate the same structure throughout the interleaver, such that when erasures occurred, some computationally intensive pre-computations can be done once, like calculating the erasure-locator polynomial [10] or inverting the generator matrix [4] or parity-check matrix [8], and used throughout for all the codewords in the interleaver.

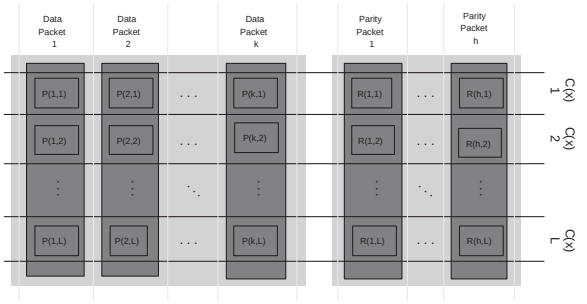


Fig. 2. Interleaving

In the next section, we will combine two decoders to get a decoder that proves to be quite efficient in some scenarios.

IV. A Combined Erasure Decoder

The ideas introduced by Berlekamp [9] can be extended to non-binary cyclic codes as well. The crux of Berlekamp's basis vector transformation decoding is that there exists several generating matrices (or sets of basis vectors) for a cyclic code $C(x)$. When erasures occurred, the erasures can be computed by using the correct set of basis vectors. Decoding erasures using basis vectors can be quite efficient as only a few additions and multiplications are needed per erased symbol. However, to find the correct set of basis vectors for the non-binary case, i.e. codes over $GF(2^m)$, remains a non-trivial and time consuming task.

With a proper designed interleaver, as discussed in the previous section, decoding using basis vector transformations can be done quite efficiently. Here, only one set of basis vector transformations needs to be computed for the whole interleaver.

The format of the decoding basis vector matrix will be determined by the positions of the erasures. The decoding matrix will have the same dimensions as a normal generating matrix. The (n, k) decoding matrix consists out of k codewords from $C(x)$. These k codewords can be computed by using an existing erasure decoder. Again, when considering the basis vector transformations, only a fixed set of coefficients are unknown, and by using pre-computations the decoding process can be optimized.

For each received column, there will be exactly one codeword in the basis vector set with ones corresponding to the received coefficients of that column. The rest of the codewords will have zeros at the corresponding coefficients. The erased symbols will be marked as erasures in all the codewords of the set of basis vectors. As an example, consider a $(7, 4)$ cyclic code used with the interleaver with 1 symbol per column. Assume that the first, fourth and sixth packet where lost. The corresponding decoding matrix will have the form:

$$\begin{bmatrix} b_{1,1} & 1 & 0 & b_{4,1} & 0 & b_{6,1} & 0 \\ b_{1,2} & 0 & 1 & b_{4,2} & 0 & b_{6,2} & 0 \\ b_{1,3} & 0 & 0 & b_{4,3} & 1 & b_{6,3} & 0 \\ b_{1,4} & 0 & 0 & b_{4,4} & 0 & b_{6,4} & 1 \end{bmatrix}, \quad (1)$$

where the elements $b_{i,j}$ are elements in the $GF(2^m)$ field.

After the above codewords have been decoded, the basis vectors can be used throughout the interleaver.

A major advantage when basis vector transformations are used is that no syndrome computation is needed. With decoders using syndrome computation, a syndrome needs to be calculated for each codeword in the interleaver, which can become quite time consuming.

In the following section we will compare the performance of the various decoders for various scenarios.

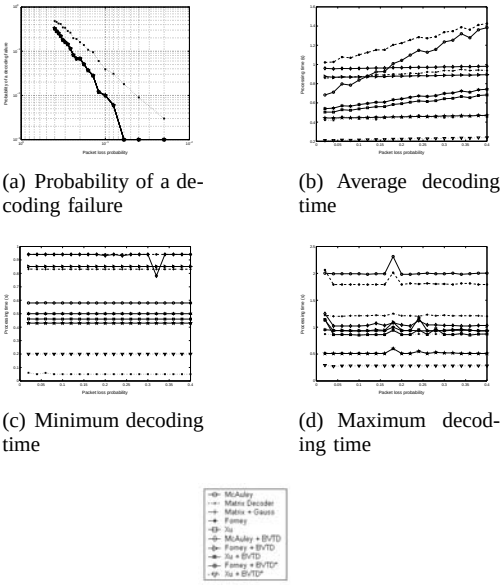
V. Decoder performance

In graphs 3, 4, 5 and 6, we present the performance of McAuley's decoder [7], two decoders introduced by Versfeld et al. [15] (Matrix Decoder and Matrix + Gauss), the decoder mentioned in Kamali's paper [10] based on Forney's algorithm (Forney), the decoder mentioned by Xu et al. [8] (Xu), Basis vector transformations with McAuley's method used to calculate the basis vectors (McAuley + BVTD), Basis vector transformations with Forney's algorithm used to calculate the basis vectors (Forney + BVTD), Basis vector transformations with Xu's decoder used to calculate the basis vectors, the optimized basis vector transformation decoder with Forney (Forney + BVTD*) and the optimized basis vector transformation decoder with Xu (Xu + BVTD*) using the interleaved structure discussed in section III.

In Graph 3, the decoders are compared for a packet erasure channel of probability ρ and packet erasure code $(5, 3)$, with an interleaver depth of 20. Each cell of the interleaver consists of three symbols. A $(15, 9)$ Reed-Solomon code was used to encode the rows. (Each packet had the same length, thus, two redundant packets were calculated and transmitted separately for three data packets). Processing time was measured when decoding started until decoding ended. When the channel introduced more packet losses than can be recovered, all the decoders ignored the data. Thus, the average processing time is calculated only for the attempts where enough packets were received.

In Graph 4, the same setup was used as above except that the interleaver depth was increased to 40. In Graph 5, the decoders are compared for a packet erasure channel of probability ρ and packet erasure code $(15, 12)$, with an interleaver depth of 20. Each cell of the interleaver consists of one symbol. A $(15, 12)$ Reed-Solomon code was used to encode the rows. In Graph 4, the same setup was used as in Graph 5, except that the interleaver depth was increased to 40.

Referring to Fig. 3(a), Fig. 4(a), Fig. 5(a) and Fig. 6(a), we see that all the decoders have the same probability of



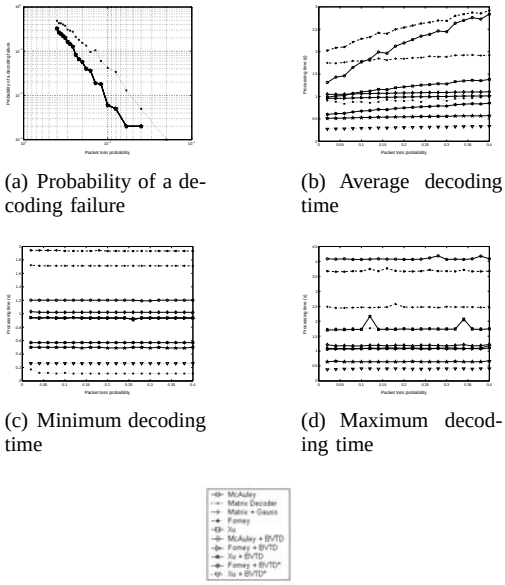
(a) Probability of a decoding failure

(b) Average decoding time

(c) Minimum decoding time

(d) Maximum decoding time

Fig. 3. Comparison of different decoders for a $(5, 3)$ packet erasure code, interleaving depth = 20



(a) Probability of a decoding failure

(b) Average decoding time

(c) Minimum decoding time

(d) Maximum decoding time

Fig. 4. Comparison of different decoders for a $(5, 3)$ packet erasure code, interleaving depth = 40

a decoding failure, except the Matrix Decoder. The Matrix decoder is in essence an erasure trapping technique, thus can only decode burst erasures contained in $n-k$ symbol positions, explaining the weaker performance.

When considering the average decoding time of the decoders (Fig 3(b), Fig 4(b), Fig 5(b) and Fig 6(b)), we see that the overall best performance is gained by using the basis vector transformation decoder where the basis vector transformations are calculated using the optimized Xu decoder. Another observation is that there is a huge increase in processing time in the decoders without basis vector transformations when the interleaver depth is increased, in contrast with the small increase in the decoding time for decoders that make use of the basis vector transformation techniques.

For the maximum decoding delay, we get that the best performance is again gained by the basis vector transformation decoder where the basis vector transformations are calculated using the optimized Xu decoder. In all the cases, this decoder is followed by the basis vector transformations decoder where the basis vector transformations are calculated using the normal Xu decoder. For the minimum decoding delay, we get that the Matrix Decoder outperform the other decoders for some cases. This usually happen when no erasures are present in the first or last symbol positions and all the erasures are contained within $n-k$ symbols. When erasures of both the first and last symbols occurred and all the erasures are contained in $n-k$ positions, i.e. a wrap around burst, the data should be shifted first. This shift then introduces delay, which then also explains the difference between the minimum delay and maximum delay of this decoder. The basis vector transformation decoder where the basis vector transformations are calculated using the optimized Xu decoder has the second best performance for all the cases. For small cell sizes, i.e. possible small number of erasures, McAuley's decoder performs the third best.

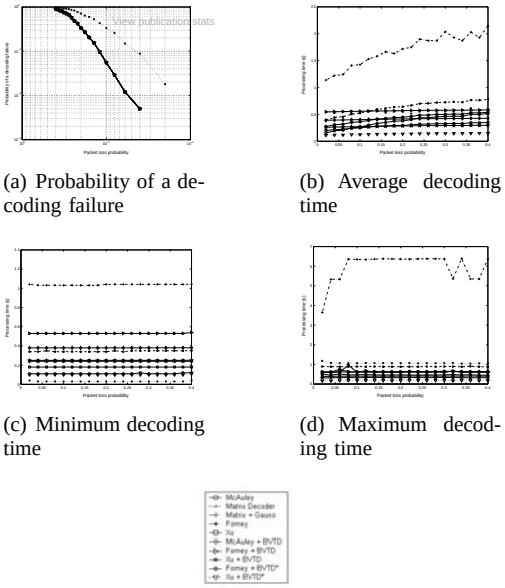
VI. Simulation Results

A simulation was done to simulate the scenario depicted in Fig. 1. Java was used to do the simulation. Five constant bit rate sources were created. The data from the sources were sent to a FEC encoding agent. The FEC encoding agent appended a FEC protocol header to each packet. Also, a redundant packet was calculated and sent for each group of five packets. A bursty background traffic generator was used to simulate Internet traffic.

The bandwidth limited router was simulated by two buffers. Data was being added to the incoming buffer. A thread was continuously checking whether there was space in the outgoing buffer to add packets. Whenever the capacity of the outgoing buffer was exceeded, the incoming buffer's contents were cleared. The outgoing buffer was serviced every x seconds in order to simulate a bandwidth limited router.

At the FEC decoding agent, copies of the data packets were stored, while the packets were forwarded to the next segment of the network. Each block of data packets received a timestamp. When the timestamp was exceeded by a certain threshold, decoding could start if any of the data packets was lost, and enough packets were received to enable decoding. If the timestamp of a block was exceeded by a maximum threshold, the whole block was discarded.

Due to processing limitations, rather slow rates were chosen for the simulation. Each constant bit rate source transmitted at a rate of 1 packet per 75 seconds. The background traffic generator transmitted between 0 and 10 packets every 3 seconds. The outgoing buffer was serviced every second, i.e., one packet was forwarded every second. The queue length was chosen as 23.



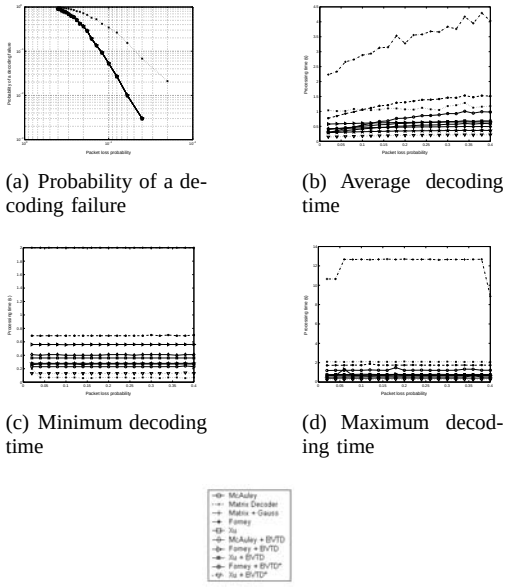
(a) Probability of a decoding failure

(b) Average decoding time

(c) Minimum decoding time

(d) Maximum decoding time

Fig. 5. Comparison of different decoders for a $(15, 12)$ packet erasure code, interleaving depth = 20



(a) Probability of a decoding failure

(b) Average decoding time

(c) Minimum decoding time

(d) Maximum decoding time

Fig. 6. Comparison of different decoders for a $(15, 12)$ packet erasure code, interleaving depth = 40

With the setup as above, the following results were obtained.

Host	Dropped %	Without FEC %	With FEC %
0	0.09	0.91	0.96
1	0.09	0.91	0.91
2	0.09	0.91	0.96
3	0.09	0.91	1.00
4	0.00	1.00	1.00

VII. Conclusion

Packet loss can degrade the QoS of real-time data substantially. Packet loss occurs due to congestion errors on normal networks and due to contention errors on optical networks. An important factor when using FEC schemes to recover lost packets of real-time data is the time in which the lost data can be recovered, as data that is delayed too long can be regarded as lost. In this paper the authors proposed and simulated a FEC enabled network protocol to compliment current techniques like LLQ to combat congestion errors on normal packet-switched networks. An erasure decoder were developed by combining two erasure decoding techniques. The performance of various erasure decoders were then compared. These techniques can also be deployed in optical networks to alleviate the effect of packet loss due to contention errors. Another possible application of these techniques can be in RAID-like disc systems.

References

[1] J. Blömer, M. Kalfane, R. Karp, M. Karpinski, M. Luby, and D. Zuckerman, "An XOR-based erasure resilient coding scheme," International Computer Science Institute Berkeley, California, Tech. Rep., 1995. [Online]. Available: citeseer.ist.psu.edu/84162.html

[2] V. Vokkarane, J. Jue, and S. Sitaraman, "Burst segmentation: An approach for reducing packet loss in optical burst switched networks," in *Proc. IEEE International Conference on Communications (ICC 2002)*, April - May 2002, pp. 2673–2677.

[3] C. Perkins, O. Hodson, and V. Hardman, "A survey of packet loss recovery techniques for streaming audio," *IEEE Network*, vol. 12, pp. 40–48, 1998.

[4] L. Rizzo, "Effective erasure codes for reliable computer communication protocols," *ACM Computer Communication review*, vol. 27, pp. 24–36, Apr. 1997.

[5] J. Bolot, S. Fosse-Parisis, and D. Towsley, "Adaptive FEC-based error control for interactive audio on the internet," in *IEEE Infocom '99*, pp. 1453–1460.

[6] V. Hardman, M. A. Sasse, M. Handley, and A. Watson, "Reliable audio for use over the Internet," *Proceedings of INET, Oahu, Hawaii*, 1995. [Online]. Available: citeseer.ist.psu.edu/hardman95reliable.html

[7] A. J. McAuley, "Reliable broadband communication using a burst erasure correcting code," in *Proc. ACM SIGCOMM '90; (Special Issue Computer Communication Review)*, Sept. 1990, pp. 297–306, published as Proc. ACM SIGCOMM '90; (Special Issue Computer Communication Review), volume 20, number 4.

[8] Y. Xu and T. Zhang, "Variable shortened-and-punctured Reed-Solomon codes for packet loss protection," *IEEE Transactions on Broadcasting*, vol. 48, pp. 237–245, 2002.

[9] E. Berlekamp, "Long block codes which use soft decisions and correct erasure bursts without interleaving," in *Proc. of the National Telecommunications Conference*, Los Angeles, USA, 1977, pp. 1–2.

[10] B. Kamali and P. Morris, "Application of erasure-only decoded Reed-Solomon codes in cell recovery for congested atm networks," in *Vehicular Technology Conference, 2000*, pp. 982–986.

[11] M. Arai, A. Yamaguchi, and K. Iwasaki, "Method to recover Internet packet losses using (n, n-1, m) convolutional codes," in *Proceedings International conference on dependable systems and networks*, 2000, pp. 382–389.

[12] N. Alon and M. Luby, "A linear time erasure-resilient code with nearly optimal recovery," *IEEE Transactions on Information Theory*, vol. 42, pp. 1732–1736, Nov 1996.

[13] M. Luby, "LT codes," in *Proceedings of the 43rd Symposium on Foundations of Computer Science (FOCS-02)*. Los Alamitos: IEEE COMPUTER SOCIETY, Nov. 16–19 2002, pp. 271–282.

[14] I. Cidon, A. Khamisy, and M. Sidi, "Analysis of packet loss processes in high-speed networks," *IEEE Transactions on Information Theory*, vol. 39, pp. 98–108, Jan. 1993.

[15] D. Versfeld, H. Ferreira, and A. Helberg, "A Reed-Solomon decoding algorithm for correcting bursts of erasures in real-time data in packet switched networks," in *Proc. 2003 IEEE Information Theory Workshop (ITW'2003)*, Paris, France, 2003, pp. 58–61.