Hierarchical Encoding of MPEG Sequences Using Priority Encoding Transmission (PET)

Christian Leicher
The International Computer Science Institute
Berkeley,CA
&
Lehrstuhl für Kommunikationsnetze
Technische Universität München

TR-94-058
November 1994

ABSTRACT

Network heterogeneity has become a major issue and multimedia applications have to cope with interconnected networks consisting of many subnetworks of unevenly distributed resources. Real-time traffic caused by video sources is bursty by nature, resulting in buffer overflow at the switch and unavoidable packet loss. Therefore it is desirable for information to be compressed and prioritized in a way that the application degrades gracefully under adverse network conditions.

Priority Encoding Transmission (PET) is a new approach to the transmission of prioritized information over lossy packet-switched networks. The basic idea is that the source assigns different priorities to different segments of data, and then PET encodes the data using multi-level redundancy and disperses the encoding into the packets to be transmitted. The property of PET is that the destination is able to recover the data in priority order based on the number of packets received per message.

This work addresses the hierarchical encoding of MPEG video streams in a PET scenario. Its focus is more on the recovery aspect, rather than on computational issues. The basic idea is that inter-frames are less redundantly encoded than intra-frames. It introduces a scenario which should prove the feasibility of our design considerations and describes simulation results with different MPEG sequences.

ACKNOWLEDGEMENTS

My six-month stay in Berkeley has been an unforgettable once in a lifetime experience. I felt very proud and fortunate to be able to work in this excellent research environment at the International Computer Science Institute. I am grateful to everybody who supported my studies, in particular:

- Prof. Ferrari who offered me the unique opportunity to write my thesis at the International Computer Science Institute and for his encouragement and support throughout my studies.
- Prof. Eberspaecher for believing in my ability to accomplish my project abroad. Without his guidance and patience this thesis would not have been possible.
- The PET team, Dr. Andres Albanese, Prof. Mike Luby, Dr. Johannes Bloemer, Dr. Jeff Edmonds and Malik Kalfane, a wonderful group of people I felt very fortunate to associate with. Their company and friendship have made my life and research here much more enjoyable. I felt very proud to be a member in their team. Their invaluable advice will benefit my future career for a long time. For their future projects I wish them continuing success.
- Dipl.Wirtsch. Inf. Wieland Holfelder for his never ending patience and friendship. Thanks to his support I acquired a lot of essential skills and last but not least survived the FrameMaker jungle.
- all members of the Tenet group and staff of ICSI who I really enjoyed working with. In particular Renee Reynolds for being a great help in organizational matters.

Most of all I am deeply indepted to my family and Carolin. Without their love, never ending support and encouragement throughout my studies this thesis would never had been possible.

Table of Contents

List of Figures

List of Tables

1 Introduction

Multimedia computing enables new applications such as Video-on-demand, multimedia email, documents and spreadsheets and desktop video conferencing. Being a powerful tool it will have a dramatic impact on social, educational and business communications:

"The transition to distributed multimedia applications will be more significant than the transition to graphical user interfaces because it will have a greater impact on business productivity and our personal lives" [Row94]

By introducing optical fiber to transmit data a tremendous increase in network bandwidth has been achieved, allowing widespread use of multimedia applications. Wide area networks of global dimensions offer ubiquitous access to users not only in the research community but increasingly to businesses. Therefore network heterogeneity has become a major issue and applications have to cope with interconnected networks consisting of many subnetworks. Computational power, bandwidth, storage and congestion control policies may unevenly vary from network to network resulting in unavoidable congestions, delays and losses [AlFe94,Part94].

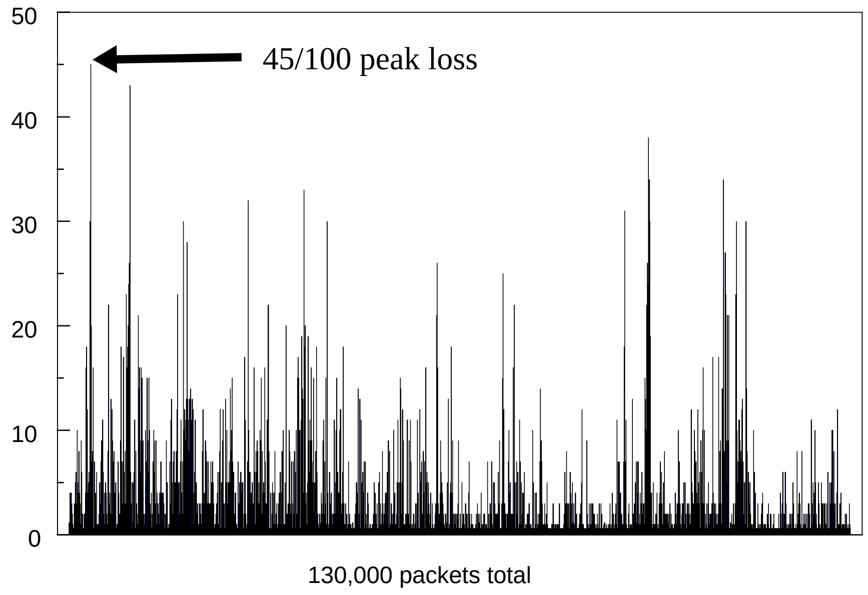

Figure 1: Packet loss per 100 observed during a multicast session over the MBone

Figure 1 shows the packet loss per 100 measured at one receiving host during a network video (nv) multicast session over the Multicast Backbone testbed (MBone) from Palo Alto to Berkeley. Note that the peak loss is ten times greater than the average loss of 4.5%.

The transmission of images, moving or still, requires an enormous amount of bandwidth. Efficient compression algorithms have been developed but still network traffic will easily be dominated by video. Contrary to computer data traffic, a set of mechanisms is required in order to provide good quality of service (QOS) guarantees. Strong delay constraints challenge the design and engineering of communication systems. Circuit switching technology imposes constant bit-rate (CBR) utilization of the assigned channels. Thus video encoders are forced to generate a CBR stream, paying the price of varying picture quality and dissatisfying bandwidth exploitation[Part94,Ga93]. Packet-switched networks such as the Internet and wide area ATM networks provide a basis for variable bit-rate (VBR) video transmission and constant picture quality. In ATM networks VBR coding schemes may be implemented without sacrificing bandwidth utilization efficiency, since bandwidth can be allocated on demand [Ga93,PaZa93]. However, VBR coding and transport introduces other potential obstacles.

Video traffic is bursty by nature. Thus when many sources are transmitting at their peak rate the available network bandwidth may be exceeded, causing buffer overflow at the switch and packet loss. Thus, packet loss is more likely to occur in bursts, rather than be evenly distributed. Although some loss of less important visual information may be acceptable, the introduction of compression algorithms using source coding techniques and motion compensation may lead to severe degradation in terms of picture quality when losses occur. Encoded bitstreams using variable length codes (VLC) are very sensitive to losses, since the loss of a single bit forces the decoder to discard information until the following synchronization sequence.

In order to guarantee a minimum quality of service to be maintained, priority mechanisms are desirable. A widely used standard for video compression has been developed by the Moving Pictures Expert Group (MPEG) [ISO11172]. Although MPEG-1 does not provide scalability, it may also be viewed as hierarchically structured. Reference frames (intra frames) carry the complete set of parameters needed for frame reconstruction at the decoder, whereas other frames (inter frames) contain only information about changes from these reference frames [Sha92]. However, some efforts have been made adding scalability in terms of resolution to the MPEG-1 bitstream. A two layered scheme over ATM has been proposed, prioritizing packets in two levels: Base layer and enhancement layer. The base layer contains only low frequency DCT (Discrete Cosine Transform) coefficients and can be decoded independently, whereas the enhancement layer carries high resolution data that is only usable when added back to the base layer. By separating the bitstream the bit rate per frame is increased by approximately 20%. The network has to make sure that the base layer is transmitted over a more reliable channel [Pa92,PaZa94].

Several other approaches introduce network support in order to cope with packet loss. One idea is based on retransmission of discarded packets. Techniques such as Automatic Repeat Request (ARQ) have significant drawbacks, since they introduce additional roundtrip latency and in a multicast scenario cause enormous complexity [Sha92].

Alternatively, codes based on Forward Error Correction (FEC) traditionally focus both on detection/correction of errors and recovery of lost information. In a packet-based network, error detection/correction can be dealt with on a packet by packet basis. Thus, the primary use of FEC by the application would be to recover lost packets [LaEf93,Sha92].

Traditional FEC permits encoding only on a single priority level. An Adaptable Forward Error Correction (AdFEC) scheme has been proposed [LaEf93], capable of assigning different priority levels to different groups of packets. The more important the group of packets, the more redundancy is devoted.

Priority Encoding Transmission (PET) [AlLu94] is a new approach regarding the prioritized transmission of scalable messages over lossy packet-switched networks. It allows a user to specify priorities for different segments of data. Based on this hierarchical structure the sender uses the system to encode the data into packets for transmission. The idea is that the information is distributed over all the packets sent per message. However, each packet contains relatively more information about the higher priority segments of the data. Hence the receiver is able to recover the information in priority order, based on the number of packets received per message.

This work addresses the hierarchical encoding of MPEG-1 video streams in a PET scenario. In Chapter 2 we describe a PET system using erasure coding techniques based on properties of polynomials over a finite field. Chapter 3 is aimed at providing basic information about the MPEG-1 standard. Due to the intention of our work, the focus is more on the bitstream hierarchy, rather than on coding techniques. Using the proposed PET scheme, Chapter 4 merges PET and MPEG. We describe a way of packetizing MPEG-1 bitstreams. The basic idea is that inter-frames are less redundantly encoded than reference frames. We introduce a PET-MPEG-1 simulation and describe observed results with different MPEG sequences.

2 Priority Encoding Transmission (PET)

Priority Encoding Transmission (PET) [AlLu94] is a new approach introducing a method for sending messages over a lossy packet-switched network according to a user specified prioritization scheme. It focuses on fault tolerant transmission, especially suitable in case of unpredictable packet losses. Its design is independent from any specific scalable application. The basic idea is that the source assigns different priorities to different segments of data resulting in a multilevel redundancy distribution. The information is distributed over all the packets sent per message. However, each packet contains relatively more information about high priority data. Hence the destination is able to recover the information in priority order based on the amount of packets received per message. In this chapter we describe a PET system using erasure coding techniques based on polynomials over a finite field.

In the first subsection we briefly introduce the notion of erasure codes. The second part describes basic properties of polynomials over finite fields, whereas the third section introduces a possible implementation of a PET system.

2.1 Erasure Codes

Let us assume a message M consists of b words of length w each. Consequently the message length m adds up to m = wb. Message M is redundantly encoded into code E(m) with length e = nw, where $n \geq b$. E is called an erasure code if the original b words can be reconstructed from any b words of the encoded message E(m) (together with the indices of the b words of encoding).

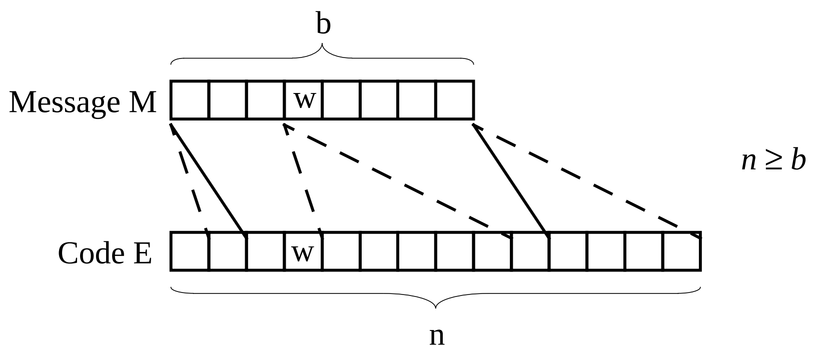

Figure 2: Erasure codes

Figure 2 displays a message M and the corresponding code E. The b words can be recovered from any subset of b words of the total encoding. Two examples depicted by the continuous/dashed line respectively, are shown in Figure 2.

A possible implementation of erasure codes consists in viewing the b words as the coefficients of a polynomial of degree b-1 over a finite field. This method is described in the following sections.

2.2 Polynomials over a Finite Field

2.2.1 Fundamentals

There are two different types of finite fields or Galois fields (GF): Prime fields GF[p] and extension fields GF[p^m] where "p" denotes a prime number and "$m \geq 2$" a positive integer. Respectively they consist of $q = p$ or $q = p^m$ elements, which in the latter case can be considered as polynomials of degree m-1 with coefficients $a_i \in \mathrm{GF}[p]$:

$$a_0 + a_1 x + a_2 x^2 + \ldots + a_{m-1} x^{m-1}$$

In Galois field GF[p] operations are calculated modulo p, whereas in Galois field GF[p^m] an irreducible polynomial p(x) is needed, so calculation is done modulo p(x) [Ha93]. The arithmetic in extension fields is more complicated than in prime fields. Therefore an erasure code implemented

in a prime field was considered to be more suitable for our purposes.
One way to realize erasure codes consists in viewing the b words of message M as the coefficients of a polynomial of degree b-1 over a suitable chosen field. Code E then is built up by n polynomial values evaluated at different field elements. Hence, from any subset of b polynomial values (and the corresponding field elements at which they are evaluated) the original message can be determined by interpolation.

As an example let us assume that three integer values (2,6,3) should be transmitted:

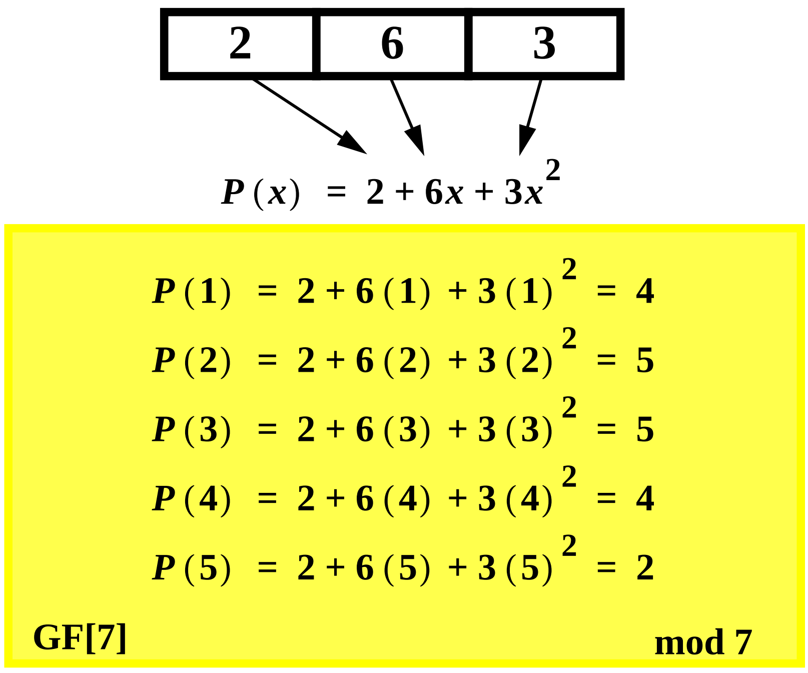
2
6
3
$P(x) = 2 + 6x + 3x^2$
$P(1) = 2 + 6(1) + 3(1)^2 = 4$
$P(2) = 2 + 6(2) + 3(2)^2 = 5$
$P(3) = 2 + 6(3) + 3(3)^2 = 5$
$P(4) = 2 + 6(4) + 3(4)^2 = 4$
$P(5) = 2 + 6(5) + 3(5)^2 = 2$
GF[7]
mod 7

Figure 3: Polynomials over Galois field GF[7]

Since all three values are less than 7, we can consider them as elements of Galois field GF[7]. Now the triplet (2,6,3) denotes the coefficients of the polynomial P(x) mod 7. It is obvious that from any three values P(i) evaluated at different field elements i the original polynomial P(x) can be reconstructed by interpolation. The redundancy is set by transmitting more than three pairs (i,P(i)). Consequently in Galois field GF[7] a maximum of seven different pairs might be created. Note that all calculations are done modulo 7.

2.2.2 A possible Galois Field for a PET System

Assuming the system uses words of length w = 16 bit or 2 bytes each, they might be considered as coefficients of a polynomial over the extension field $GF[2^{16}]$. However, from a computational point of view it is preferable to use a prime field. The Galois field $GF[2^{16}+1]$ is a prime field covering the range that can be represented by 16 bit words. Multiplications within this field can be reduced to a constant number of integer arithmetic operations, shifts and comparisons. However, the value 2^{16} is not representable by 16 bits, but obviously may be one of the polynomial values evaluated at different field elements. Therefore in order to avoid overflow a PET system has to provide an additional feature.

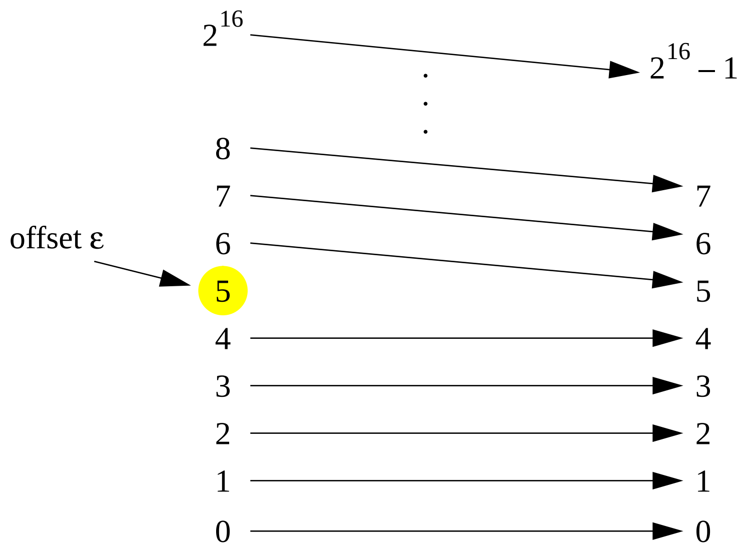

Figure 4: Representation of Galois field $GF[2^{16}+1]$

In case the list of transmitted polynomials contains the element 2^{16}, an offset ε has to be defined which makes sure that the range representable by two bytes is not exceeded. The offset per packet is determined by browsing the transmission data for a value which is not a member of the data within the same packet. Then all values smaller than the offset $x < \varepsilon$ are mapped onto themselves $x = x$ and all values $x > \varepsilon$ are mapped onto $x = x-1$. Consequently within each packet, 16 bits have to be reserved for the offset.

2.3 Basic Design Considerations

2.3.1 Data Partitioning

The first step in constructing a PET system is partitioning the transmission data. These portions are the basic units, called messages which are encoded one at a time. A message then is split into user specified priority segments. A segment is divided into blocks. Each block represents a polynomial of degree (block length) - 1. Within each segment all blocks have to have the same length in order to meet the given priority. Then each block is encoded separately by using erasure codes. The n polynomial values of the erasure code E(m) are dispersed into n packets. Each packet therefore contains only one polynomial value represented by a single word. The field element varies from packet to packet, but within a single packet all polynomials are evaluated at the same field element.

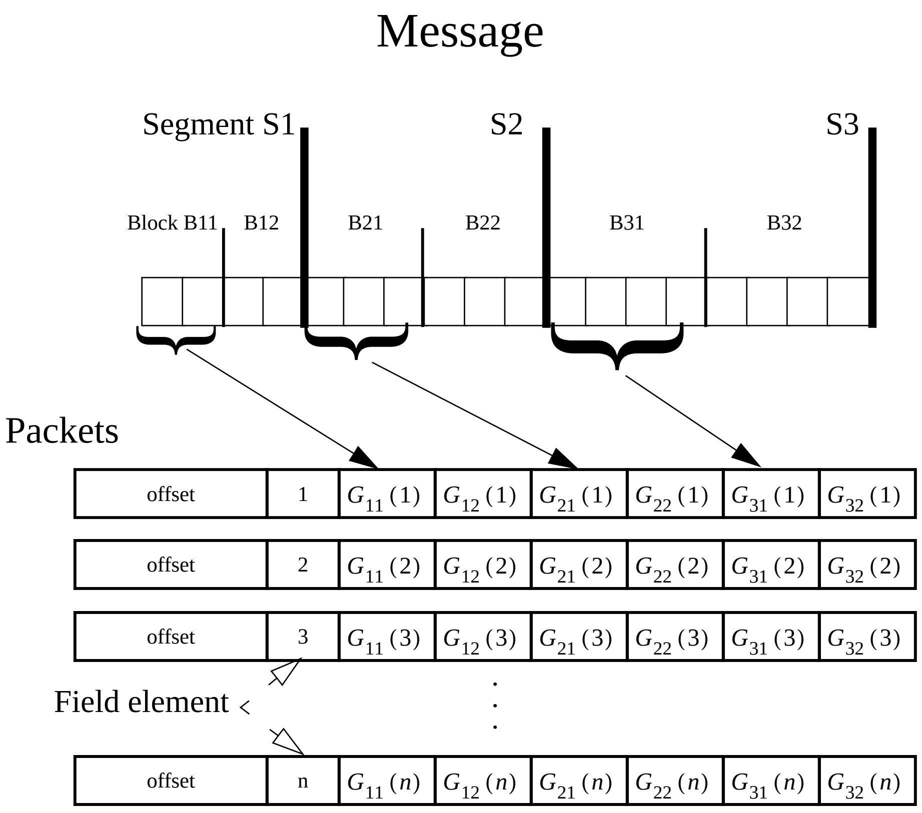

Figure 5: Message striping process

Figure 5 shows an example message partitioned into three priority segments and striped into packets of equal length. Note that the maximum number of packets sent is $2^{16} + 1$, the size of the Galois field GF[$2^{16} + 1$].

The smallest entity is a word w of two bytes. The priority of a segment is expressed by the length of its blocks. Hence the smaller the blocks, the higher the priority. Obviously in our sample message segment S1 has the highest priority, followed by S2 and S3. Consequently blocks B11-B12 consist of two, B21-B22 of three and B31-B32 of four words. $G_{sb}(i)$ denotes the value of the corresponding polynomial of degree equal to (block size) - 1, evaluated at the field element i (indices s and b identifiy the segment and block respectively). Within each packet all polynomials are evaluated at the same field element. Hence B11-B12 can be reconstructed from any two packets, B21-B22 from any three and B31-B32 from any four packets. By sending n=6 packets B21 now would be encoded with 50% redundancy, that is it can be recovered from any 50% of the transmitted packets. Each packet additionally carries at the beginning the offset discussed above and the field element i.

2.3.2 The Priority Table

One of PET's basic design principles is that the priority function is specified by the user or application. Based on this distribution the message is divided into segments. This process is described in detail in Chapter 4. A simple and flexible realization consists in specifying priorities in fraction of packets needed to recover the segment.

Regarding the previously discussed example (Figure 5), a possible user defined table may look as follows:

Table 1: Priority table information

Segment Size (in words)	**Fraction of Packets Needed to Recover**
S1 = 4	0.333
S2 = 6	0.500
S3 = 8	0.666

With respect to Figure 4 the number of transmitted packets n is set to 6. Consequently even though segment S1 encompasses only 22.22% of the total message it covers a third of the encoding. Since any pair of packets is already sufficient to recover S1, an overall overhead of 200% for S1 is sent. The corresponding values for the other segments are listed below in priority order:

Table 2: Information distribution

Segment Size	**Fraction of Message Size**	**Transmitted Redundancy**
S1 = 4	22.22%	200%
S2 = 6	33.33%	100%
S3 = 8	44.44%	50%

The total encoding length adds up to two times the length of the original message, not included the overhead for the offset, field element and the priority table.

3 MPEG-1 Video Compression

In 1988 a group called Motion Picture Expert Group (MPEG) was formed to produce a standard for video and audio encoding for VHS-quality compression at bit-rates less than 1.856 Mbits/sec. However the standard is not limited to this benchmark and may be used at higher data rates [ISO11172]. Intentionally it was designed for storage on digital storage media such as CD-ROM and DAT which are perfectly suited for bit-rates of 1 to 1.5 Mbits/sec. Its development therefore addressed the following features [Gall91]:

- Random Access
- Fast Forward/Reverse Searches
- Reverse Playback
- Audio-Visual Synchronization
- Editability
- Format Flexibility

Image compression usually is achieved by applying several techniques such as subsampling of chrominance components, quantization, frequency transformation by cosine transform (DCT) and variable length coding (VLC)[PeMi93]. MPEG additionally introduces motion compensation (MC) to exploit temporal redundancy, predictive coding and picture interpolation.

This chapter is aimed at providing a quick overview of the MPEG standard and its features.

3.1 Coding Layers

As illustrated below an MPEG bitstream is built up of four major components [GoRo94,PaZa92].

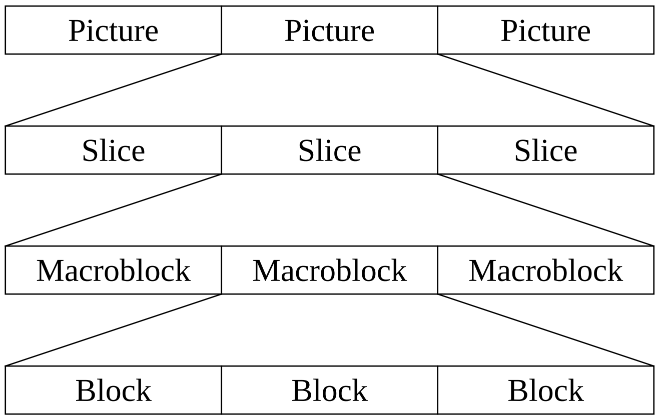

Figure 6: MPEG coding layers

Pictures are the equivalent of a single movie frame and represent the basic unit of display. There are three different kinds of pictures, which are discussed later in more detail. A frame is divided into *slices*. They are the basic processing unit and provide fault tolerant access within a single picture. Since they are independent, they may be coded in groups. A slice may be a single line of the image or consists of any number of *macroblocks*. Each of them contains a header and six component blocks [ISO11172]:

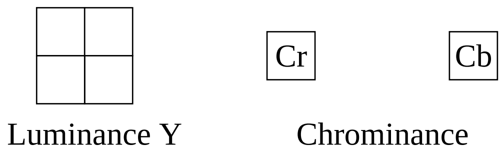

Figure 7: Subsampling of Y/Cr/Cb information

Due to the characteristics of sensitiveness of the human eye, color information is subsampled [PaSm93,PeMi93]. That is, a macroblock of 16x16 pixels represents four blocks of luminance Y, but only one block of chrominance blue Cb and one of chrominance red Cr. Squares of 8x8 pixels each are called *blocks*, the smallest coding entity in the MPEG algorithm.

3.2 Coding Techniques

MPEG uses three different types of pictures: Intra-frames (I), coded as a still image, Predicted frames (P), predicted from the most recently decoded I- or P-frame and Bidirectional-frames (B) interpolated from the closest two I- or P-frames.

An *I-frame* is encoded independently from any other image, applying techniques similar to JPEG compression: Blocks are first transformed from spatial domain into the frequency domain using Discrete Cosine Transform (DCT). The corresponding coefficients then are quantized in frequency order, that is low frequency components are encoded more accurately than high frequency ones. Additional compression is achieved by variable-length coding of the resulting data in a zig-zag ordering.

P-frames generally refer to the most recent reference frame, which is either an I- or a P-frame. They use motion compensation on a macroblock basis.

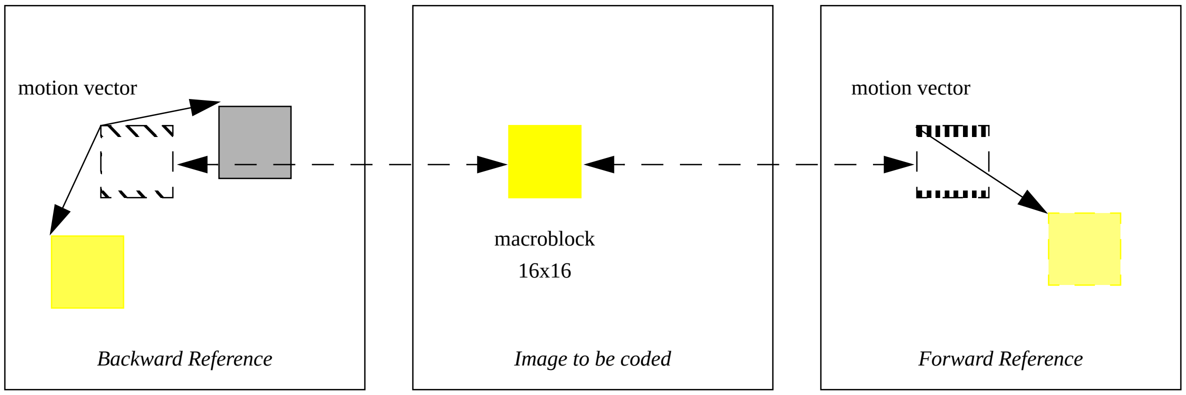

Figure 8: Motion compensation (MC) in P- and B-frames

Encoded are motion vectors and error terms. The vector specifies the relative location of the macroblock within the reference frame, which matches the one to be coded best. The difference is expressed by an error term or in case of total compliance is skipped. In the latter case the reference macroblock is simply duplicated. The range for motion vectors may be limited, since searching for the closest pattern is very time consuming. Macroblocks may also be coded as I-macroblocks.

Additionally to backward compensation, *B-frames* may be interpolated from past and future pictures. So B-macroblocks may either use backward motion compensation (MC), forward MC or both, in which case the 16x16 area is averaged (Figure 8). Bidirectional frames therefore provide the highest amount of compression, but may not be suitable for all applications, since they require out of sequence transmission which leads to latency [ISO11172].

Figure 9 shows the complete coding loop for the MPEG algorithm [PaSm93,PaZa94]. Note that the feedback loop is only needed for P- and B-frames.

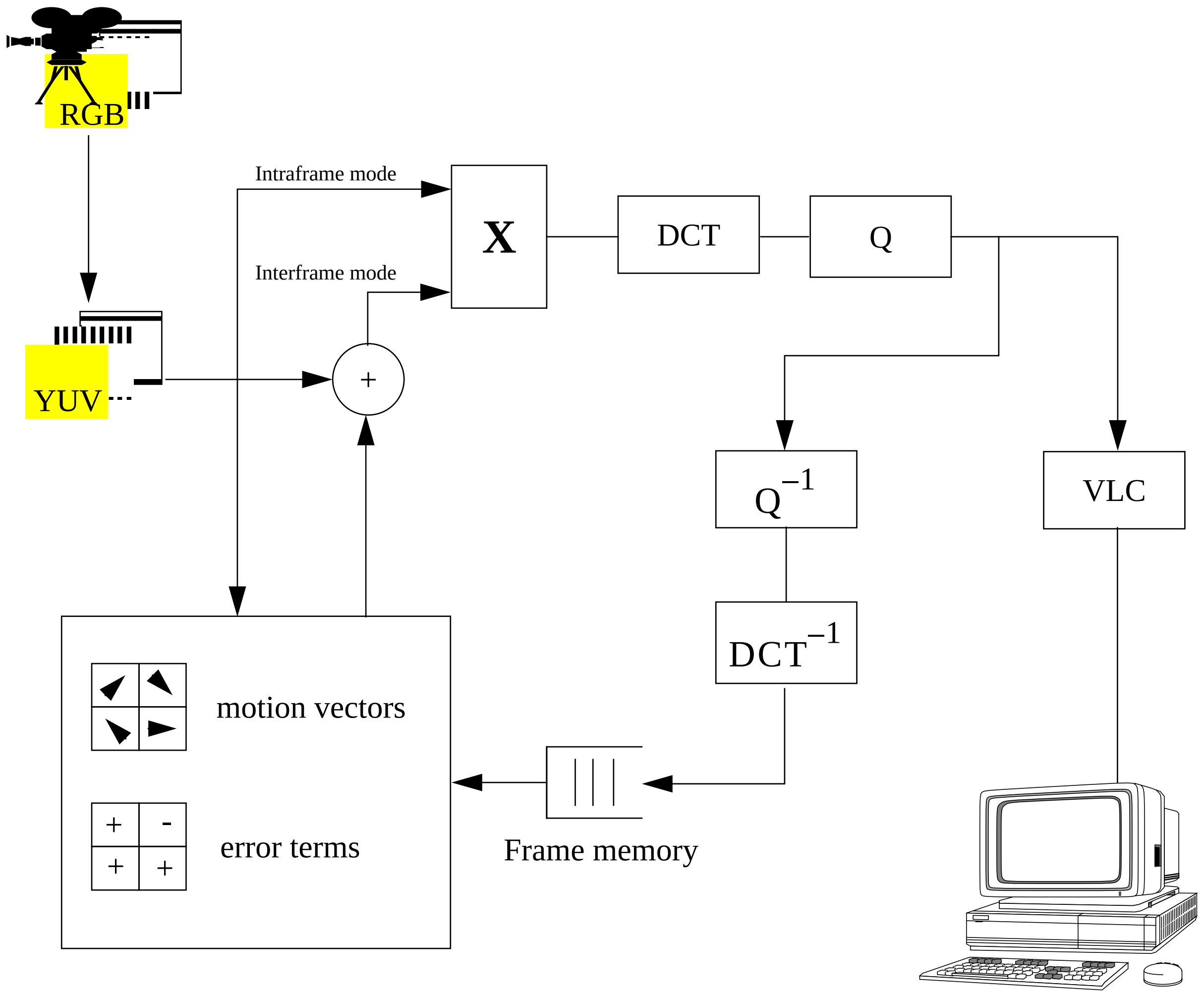

Figure 9: Main MPEG coding loop

3.3 Bitstream Hierarchy

An MPEG video is represented by a layered bitstream. The top layer is called sequence. A movie is encapsulated by sequence delimiters. It always starts with a sequence header which contains essential information such as picture size, picture rate and bit rate. Eventually it ends with a 32bit sequence end code.

The header is followed by any number of *Group of Pictures (GOP)*. A GOP provides random access, since it is the smallest coding unit which under certain circumstances can be independently decoded. Due to the variety of picture types, the design of a GOP is constrained by the following properties [ISO11172]:

- *Property 1. A group of pictures, in bitstream order must start with an I-frame and may be followed by any number of I-,P- or B-frames in any order.*
- *Property 2. It must begin, in display order, with an I- or B-frame and must end with an I- or P- frame. The smallest GOP might consist of only a single I- picture, whereas the upper bound is unlimited.*
- *Property 3. It always begins with a GOP-header and either ends at the next GOP-header or at the end of sequence.*

Note that in case a GOP starts with a B-frame in display order, it cannot be decoded independently, since it requires a reference frame of the previous GOP.

In general, a GOP visualizes the dependencies between the several types of frames within a GOP or even across their borders. As mentioned before, there are two different picture orderings: Display order and bitstream order. The latter has to be different, since for interpolation the decoder first has to know about the two reference frames, before he might evaluate the macroblocks within a B-frame slice. A simple GOP example is displayed in Figure 10:

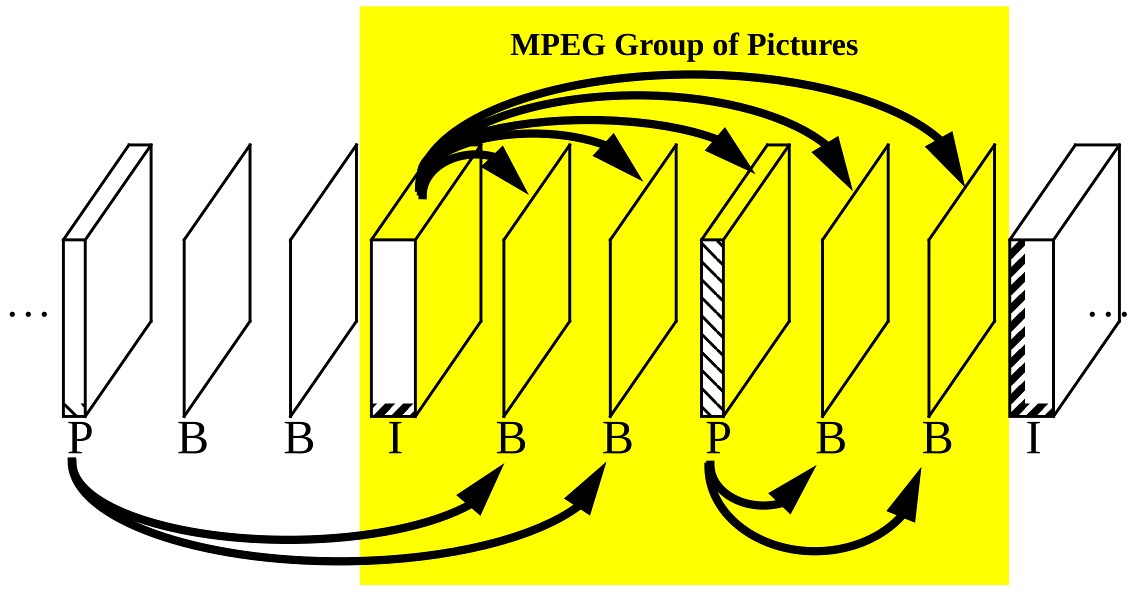

Figure 10: MPEG Group of Pictures (GOP) in bitstream order

Note that in bitstream order frames can only refer to past reference frames. In case the first block of B-frames is interpolated, it depends on the P-frame of the previous GOP. The MPEG standard provides a closed GOP flag. It denotes whether the GOP is open or closed. In the latter case a GOP can be decoded without any references to previous groups [ISO11172]. With respect to the transmission of MPEG sequences over packet-switched networks, the closed GOP flag could slightly increase the fault tolerance of the bitstream.

Figure 11 shows the difference between bitstream order and display order. Frames usually carry temporal references which denote the display order. In general bidirectional frames change places with reference frames. The very first GOP varies from the subsequent, since two references are needed in order to decode the first block of B-frames. However, the I-frameI-frameP-frame at the beginning is kind of an initialization and therefore usually is not displayed at all.

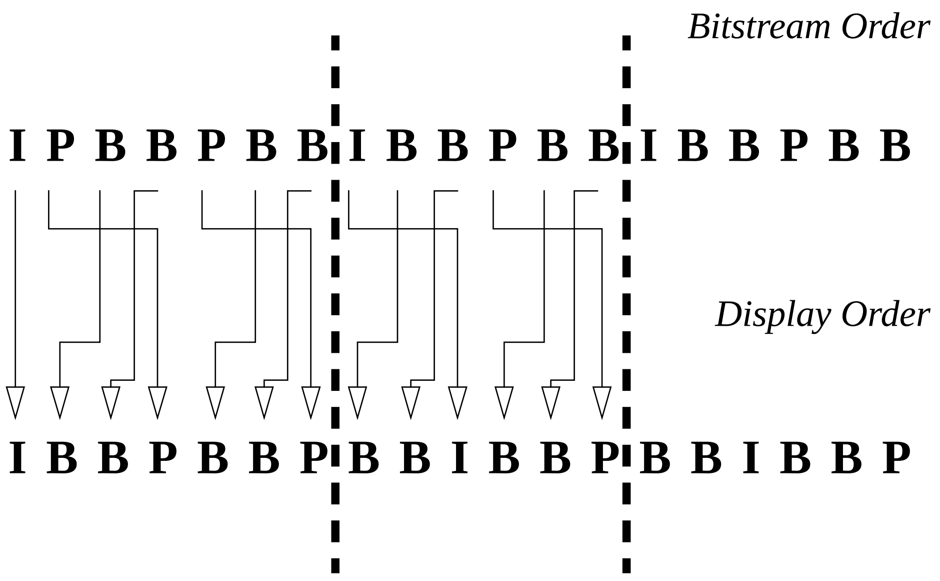
Bitstream Order
I P B B P B B I B B P B B I B B P B B
Display Order
I B B P B B P B B I B B P B B I B B P

Figure 11: Bitstream order vs. display order

4 Prioritized Encoding of MPEG Sequences

4.1 Motivation

Priority Encoding Transmission (PET) [AlLu94] is designed to be an independent interface between a packet-switched network and any scalable application. The latter has to provide a priority scheme according to which PET is able to assign redundancy. In this chapter we introduce a scenario which should demonstrate the feasibility of our design considerations and describe simulation results with MPEG sequences. The focus of this work is on the recovery of information sent over a lossy medium. It has to be pointed out once again that PET does not stick to this proposed application and is aimed at coping with any scalable data.

From an information-theoretic point of view MPEG video compression represents a source coding scheme. Even though PET does not provide error correction, it can be considered as a kind of channel encoding. Compared to systems based on Forward Error Correction (FEC) codes, PET uses less redundancy overall, since the redundancy is unevenly distributed. Concerning the network, a single channel reservation is proposed. In case the data is already layered, it should be multiplexed onto a single channel.

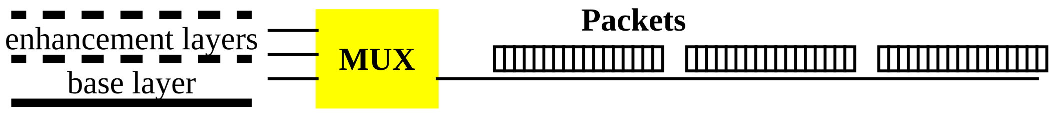

Figure 12: Multiplexing of layered data onto a single channel

Due to the rapid development of multimedia applications, there is an increasing need for video compression techniques. So far, most applications use Motion JPEG, which is considered to be more fault tolerant. Only by introducing P and B-frames, MPEG provides higher compression rates than JPEG, but at the same time makes the bitstream even more vulnerable to losses:

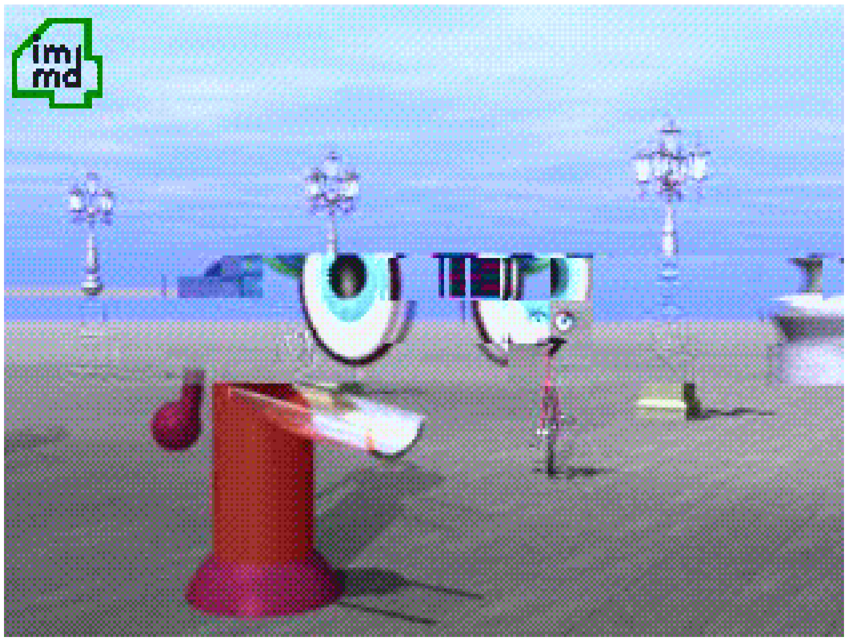

Figure 13: Losses affecting MPEG

Figure 13 displays a MPEG-1 bitstream affected by lost packets. Note how the image is mixed up from several frames. Macroblocks are not correctly replaced or use the error term in combination with an affected reference frame. Usually affected macroblocks might only vary slightly from the original, but their blocky character distorts the image.

The MPEG video compression standard is considered not to be suited for transmission over packet networks, since it was designed for storage of video. However, protected by a PET system requirements for transmission over a packet-switched networks could be met by increasing the bitstreams fault tolerance.

Video traffic is bursty by nature and therefore affects the stability of packet networks. In order to cope with losses there is an increasing demand for scalable video. Scalable extensions have been added to the succeeding MPEG-2 standard [ISO13818,ChA94], but do not promise to be very effective. MPEG-1 recently has become very popular and many software as well as hardware coders are emerging. It does not provide scalability on a picture quality basis, but the dependencies between different types of frames provide a priority scheme for PET. Since our goal was a feasibility proof of PET as an independent interface, we did not intend to change any coding techniques, such as customizing MPEG encoders. Even though the granularity in this case might be limited to a frame basis, MPEG-1 provides a good basis for a preliminary PET simulation.

4.2 Segmentation of MPEG Sequences

A MPEG video sequence is composed of sequence headers and groups of pictures (GOP). Header information usually has to be especially protected. It might be transmitted only once at the very beginning and end of a sequence. It not only identifies a bitstream as a MPEG movie, but also contains the following essential parameters set by the encoder:

- Horizontal/Vertical Picture Size
- Pel Aspect Ratio
- Picture Rate
- Bit Rate
- Constrained Parameter Flag.

As shown in Figure 10 losses within an I-frame at least affect a whole GOP and in most cases as well the first block of B-frames of the succeeding GOP. Therefore it is obvious that within a GOP an I-frame has to have highest priority. This segment should also encompass GOP header information, since without the corresponding I-frame it is of no use either. B-frames require an additional reference frame in order to be correctly decoded. This reference usually is a P-frame which itself is dependent on the previous I-frame. Consequently P-frames have to have higher priority than B-frames.

A possible way to encode MPEG is now to prioritize over a whole GOP and map it onto a PET-message. At the very beginning the message might also contain sequence header data. This solution has a certain drawback, since several frames have to be buffered first and therefore latency is added to the system. Hence in this version it might not be suited for interactive video conferencing applications, but several other scenarios such as broadcasting of conferences and Video-on-demand systems should be taken into consideration. Nevertheless in case of a small number of frames in a GOP, it might also be applicable for interactive usage.

4.2.1 Redundancy distribution

Let us consider a typical MPEG-GOP frame pattern:

I B B P B B

Regarding a picture size of 320x240 pixels, Table 3 shows representative frame sizes:

Table 3: Typical frame sizes

Frame Type	**Size in Bytes**	**Occurences in GOP**	**Total Size**	**Fraction Needed**	**Encoding**
I-frame	12,000	1	12,000	60%	20,000
P-frame	4,800	1	4,800	80%	6,000
B-frame	2,850	4	11,400	95%	12,000
Total		6	28,200		38,000

Priorities are expressed by fraction of packets needed to recover the original information. According to the above considerations, the I-frame might be encoded in a way that it can be recovered from any 60% of the total number of packets sent, the P-frame from any 80% and the two blocks of B-frames from any 95% of the packets. The very first message includes a fourth priority segment, which contains important header data. It might be encoded highly redundantly, so it can be reconstructed from any 10% of the packets.

Figure 14 displays this multilevel encoding idea. Note how the total overhead is unevenly distributed over the data. Although the original message is 74.21% the length of the total encoding, 60%, 80% and 95% of the encoding is sufficient to recover the I-, P- and B-frames respectively.

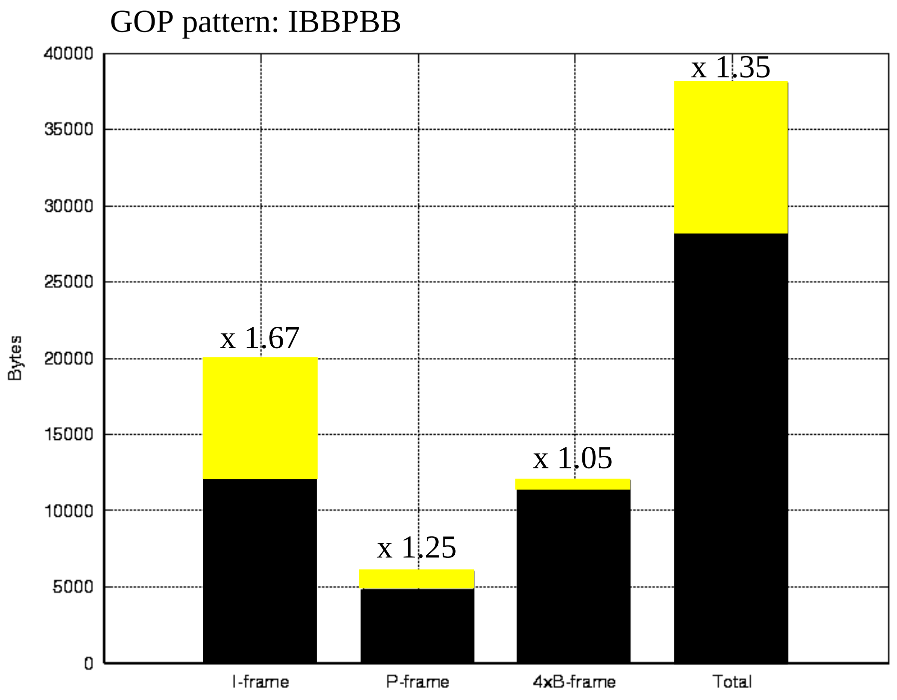

Figure 14: Multilevel redundancy distribution

4.2.2 Priority Table Setup

The priority table is to be set up by the application. It should be structured as simple as possible in order to simplify the communication process with the PET interface.

There are three major features, the interface has to know about:

- Number of priority segments

- Length of each segment
- Priority of each segment

A fixed number of 10 segments per message was considered to be enough for most MPEG groups of pictures. Each segment consists of either only header information, a single frame or a block of frames.

Priorities are inserted by fraction of packets needed to recover, multiplied by 1000. Consequently 750 corresponds to 75%. Returning to our example pattern, the priority table looks as follows:

Table 4: Priority table format

Segment Size	12,000	5,700	4,800	5,700	Not used	Not used	Not used	Not used	Not used	Not used
Priority	600	950	800	950	/	/	/	/	/	/

Note that the groups of two consecutive B-frames form one priority segment. In total 20 values are passed in a zig-zag ordering (arrow) to the interface, even though in this case only eight are needed.

With respect to mapping the group of pictures onto packets according to the priority table, we consider the following artificial scenario:

- Packet size 190 words (380 bytes)
- Total number of packets sent: 100

Packets might be considered to be UDP packets on top of an IP protocol. The total amount of packets sent is determined by the quotient of the total encoding of 38,000 bytes (Table 3) and the packet size. In order to meet the priorities specified in Table 4, the I-frame segment has to be encoded such that it can be reconstructed by any 60 packets (60% of the total amount of packets sent). Therefore the segments have to be divided into blocks of size 60 (polynomials of degree 59) for the I-frame, 80, and 95 for the other segments respectively. Figure 15 shows the striping procedure for the given example and the partitioning of. Note that this works exactly only for the given values in this example. In Section 4.5.1 we introduce a packetization algorithm that will work for any given values. For the time being we do not pay attention to the fact, how the destination knows about the table.

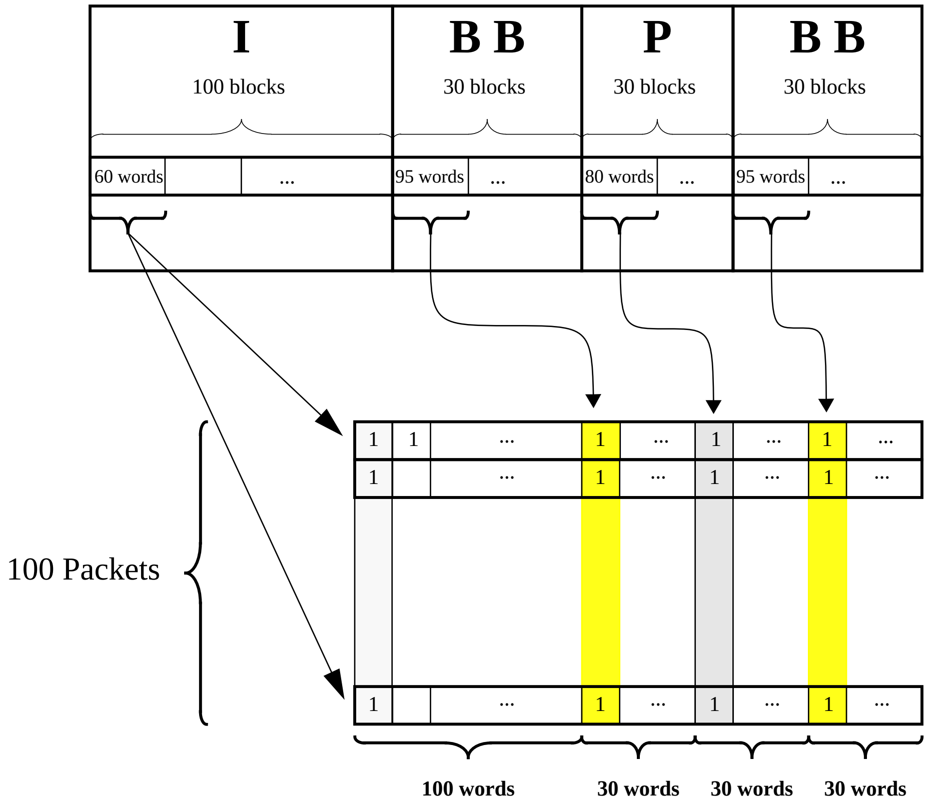

Figure 15: Mapping of group of pictures onto packets

4.3 Software Architecture

Since Priority Encoding Transmission is designed to be compatible with any scalable application or type of network, it should be implemented as an easily exchangeable interface. We introduce an architecture in which the interface consists of two modules: An application module handles the video data and passes a whole message at a time together with the priority table to a PET coding module. The latter creates the corresponding amount of packets needed and maps the message. On the decoding side, it collects the transmitted packets, recovers the information and passes the video data to the application module (Figure 16).

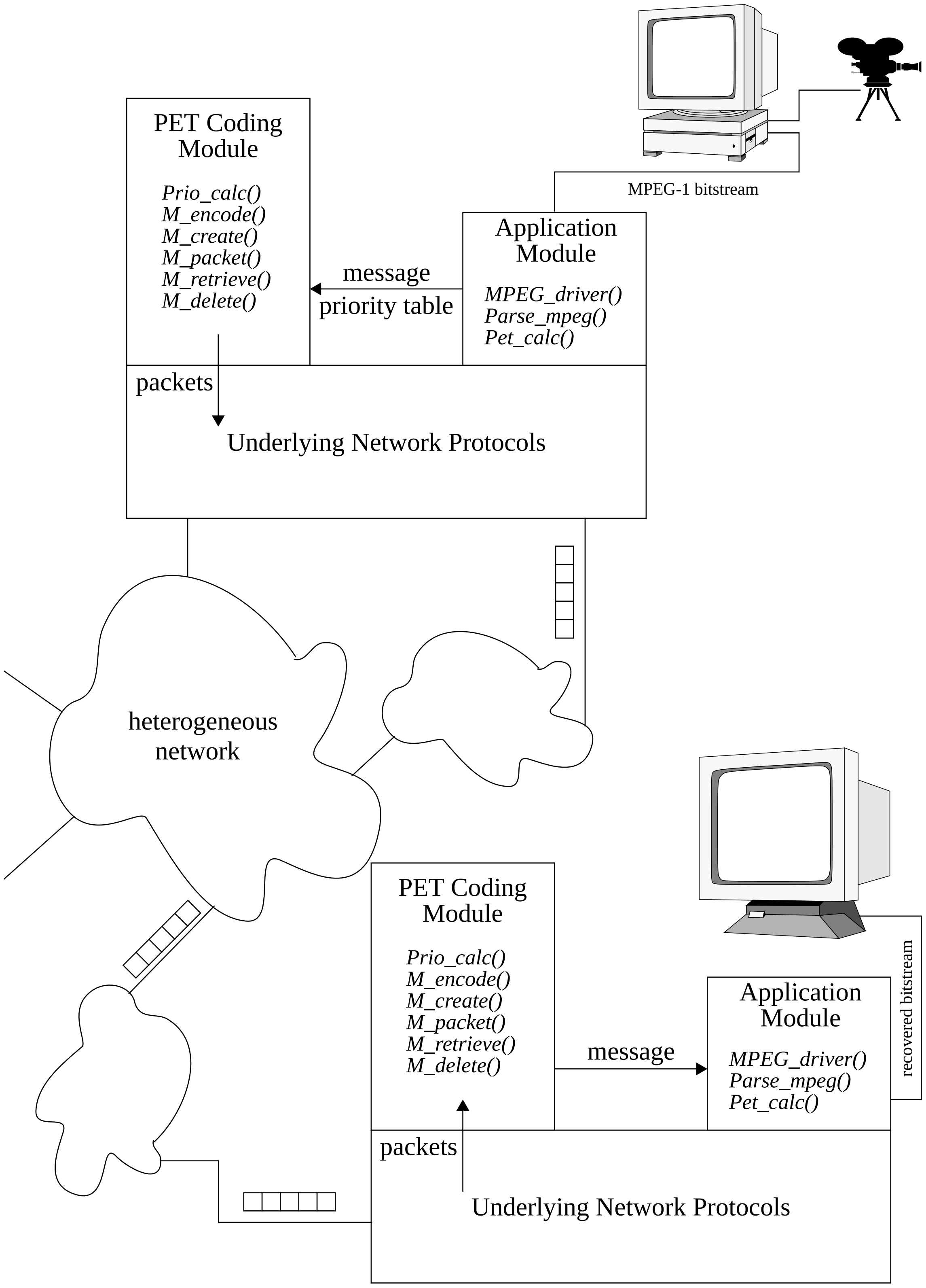

Figure 16: Software architecture

The application module consists of the following routines:

- MPEG_driver()
- Parse_mpeg()
- Pet_calc()

MPEG_driver() identifies the bitstream as an MPEG sequence and provides the user interface. The user might initialize the priority settings for the different frame types and header information. The size of the priority table could also be customized. According to the picture size and GOP length it allocates memory space for the message buffer. All other routines are invoked by MPEG_driver().

Parse_mpeg() parses the hierarchical bitstream, while reading a GOP into the message buffer. It decodes the different priority types and allocates a linked list that stores relevant data such as size, position in the buffer and number of frames about each segment.

After a message is buffered MPEG_driver() calls *Pet_calc()*. According to the user specified priorities it fills in the priority table as shown earlier in Table 4. It also has to make sure that the message is word aligned, since this is the basic processing unit of the coding routines. Therefore in case a segment boundary is not word aligned it performs a zero byte stuffing.

The message and table are then passed to the PET coding module that encompasses the following routines:

- Prio_calc()
- M_encode()
- M_create()/M_delete()
- M_packet()
- M_retrieve()

With respect to Figure 15 a kind of "internal priority table" has to be determined by Prio_calc(). Its input is the user-specified priority table, packet size and message size. The "internal table" has to have the size of a packet. Returning to the previous example its first value then would be 60, the size of the first message block and degree of the corresponding polynomial. How to fit in these values exactly into a packet and how many packets to send for a message will be discussed later on.

M_encode() then performs the actual encoding according to the internal table over the Galois field GF$[2^{16}+1]$. It evaluates the polynomials and determines the offset. Up to now we are assuming packet sizes of a couple of a hundred bytes, so we transmit the priority table uncoded with each packet. Additionally the Galois field element and offset contribute to the total overhead (Figure 5).

These are all the routines needed on the encoding side. From the destination point of view the focus is on the recovery aspect:

M_create() allocates memory for the message to be reconstructed and initializes the interpolation routines. This space later will be erased by invoking *M_delete()*.

Each arriving packet is passed to *M_packet()*. Depending on the fraction of packets received it tries to regain the original information by interpolating the polynomials. In the current version the granularity of recovering is limited to priority segments, that is at least 60% of all packets are needed to recover some useful data.

Finally *M_retrieve()* compares the amount of received packets with the priority table and accordingly erases undecodable data. For simulation reasons it also calls another routine, *create_dummy_b()*, which replaces lost frames with dummy B-frames in order to keep the same number of images in the bitstream.

4.4 Simulation Layout

In the public domain software area currently two MPEG players are available:

- MPEG-1 player (University of California at Berkeley)
- MPEG-2 player (MPEG Software Simulation Group)

The Berkeley player probably is the better known, since it has become the default tool for Mosaic. Despite its popularity it has certain drawbacks. The current version does not provide variable frame rate control. Therefore the display speed is not controllable, a feature that may be of significance in case the software decoding routines can not catch up with the ideal frame rate specified in the standard. Even more striking is that its decoding routines are not fault tolerant. Usually a segmentation fault occurs after the first error in the bitstream.

In June 1994 the MPEG Software Simulation Group has released version 1.1 of its MPEG-2 encoding and decoding routines. MPEG-2 by definition is forward compatible with MPEG-1. Consequently the player could also be used for decoding MPEG-1 bitstreams. In our simulation we decided to use this player to decode MPEG-1 sequences. The development focussed on an implementation close to the standard [MP2/94]. Therefore the performance of its decoding routines might be slightly worse than in the case of the Berkeley player, but it turned out to be much more fault tolerant. Additionally it provides frame rate control. We have customized its options in order to provide easy modification of geometry parameters and of user specified titles. At the moment it might not totally comply with the MPEG-2 standard, since a couple of features are still missing, but it definitely meets our requirements as a PET-MPEG simulation tool.

The simulation displays three MPEG movies simultaneously together with a small animation using the GNUPLOT plotting program. On the upper left side the original MPEG sequence is displayed. To the right the window titled “PET DEMO” shows the same movie. It has been packetized according to the PET scheme and packets were randomly thrown away in bursts in order to simulate a lossy medium. The packet size was set to 2000 bytes plus the priority header of 40 bytes (20 short integers). A third player simulates transmission of packets of the same size without protection. Although the simulated losses for the PET protected movie are much higher than in the MPEG sequence without protection, its performance is much better. There is absolutely no variation as far as picture resolution is concerned, whereas in the other case the picture is displayed fragmented, sometimes for up to 30 to 40 frames in a row. Especially in case reference frames are affected, sometimes a couple of rows of macroblocks are replaced by macroblocks of a picture displayed 20 frames ago. In case the MPEG + PET sequence has to sustain extremely high losses, the PET system decreases the frame rate. The effects of dropping frames depend on the GOP pattern. In case of a large number of consecutive B-frames in a row, the picture becomes jerky, whereas for example two dropped B-frames between to reference frames can hardly be realized when displayed at 20 to 30 frames per second.

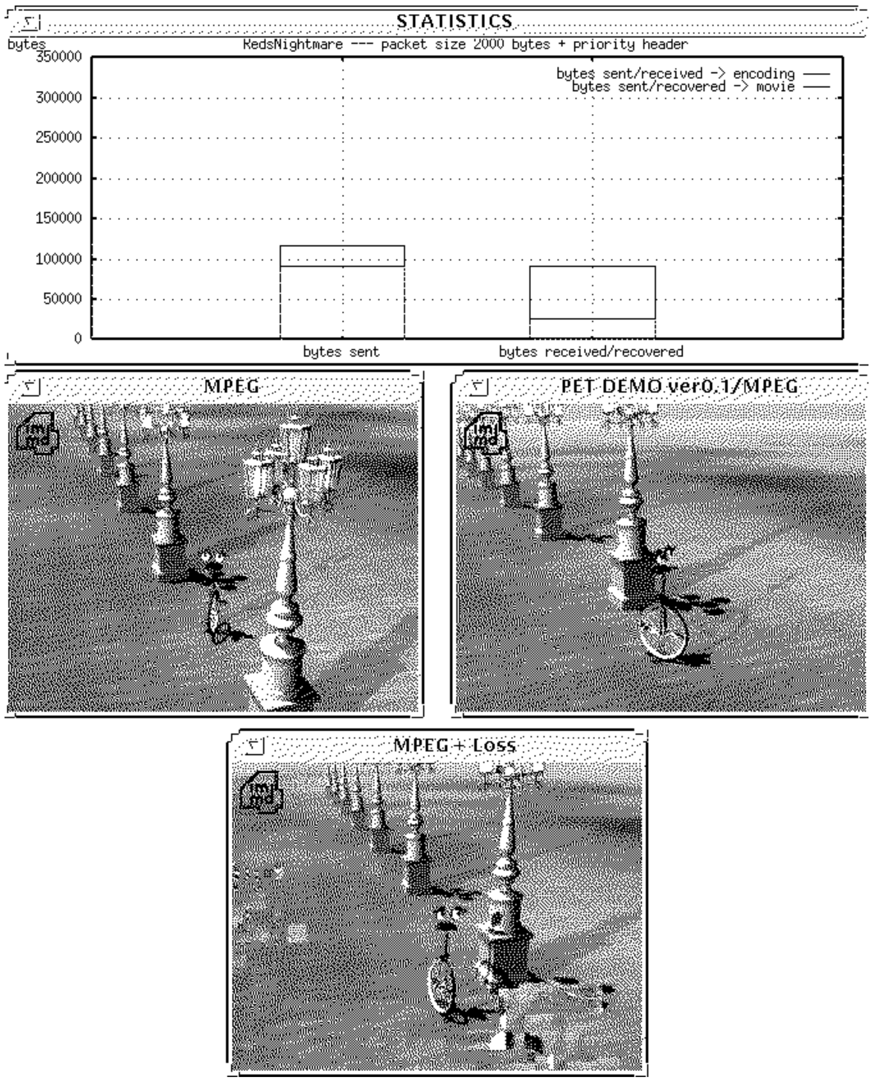

Figure 17: Simulation layout

In the statistics window there are two boxes labeled “bytes sent”. Their value is updated with every group of pictures, since packets are coded on a GOP basis. The lower, red box shows the current message size in bytes, whereas the upper, blue box represents the total amount of encoding. Consequently the gap between these boxes is a measurement for the redundancy spent per message. The two boxes to the right refer to the destination side. We distinguish between bytes received and bytes recovered. The latter corresponds to the lower, red box. In case no frames were lost, both red boxes should have the same level. The difference between the blue boxes then represents the simulated losses.

In our simulation, an exemplary MPEG video stream is used consisting of 1210 images of the animation Red’s Nightmare. The uncompressed image size was 320x240x24 which adds up to 265 Mbytes for the whole animation. The compressed MPEG equivalent encompasses 3.65 Mbytes which results in an overall compression ratio of 1:76 or a compression rate of 1.3%. Frame specific information is given in Table 5:

Table 5: Red’s Nightmare: Frame specific information

Type	**Occurences**	**compression rate**	**average size (bytes)**	**max. size**	**min. size**
I-frames	41	5.54%	12,546	26,460	7,733
P-frames	81	3.52%	8,086	27,813	495
B-frames	1,088	0.98%	2,252	17,875	290

Note that in the worst case P-frames and B-frames might be even bigger than intra coded pictures (I-frames).

The sequence is partitioned into groups of pictures (GOP) of length 30 frames each. The MPEG standard aims at a frame rate of 30 frames per second, thus a GOP would correspond to a second in the movie. Its frame pattern in the two different orderings is shown in Figure 18. Note that a lost I-frame affects at least 38 other frames, 29 of which are within the same GOP, whereas the first segment of 9 B-frames from the succeeding GOP can not be correctly decoded as well (Figure 10).

IBBBBBBBBBPBBBBBBBBBPBBBBBBBBB

bitstream order

BBBBBBBBBIBBBBBBBBBPBBBBBBBBBP

display order

Figure 18: Red’s Nightmare GOP pattern

Transmitting groups of pictures of this size definitely requires some kind of protection. On the other hand in the current version we loose 27 B-frames whenever more than 10% of the packets per message are lost (Table 6). In a future version which is currently under development, the amount of B-frames recovered gradually increases with the amount of received data.

The difference between the orderings becomes very visible in the simulation, since in case all frames but the I-frame are lost the dummy B-frames are displayed first. This gives the impression that the two sequences are not synchronized anymore which is obviously not the case.

Encoding one GOP at a time Red’s Nightmare is split into 41 messages. Due to the variation in frames sizes, GOPs range from 20,150 bytes up to 239,436 bytes. Assuming a frame rate of 30 frames per second the latter value provides the peak transmission rate of 1.82 Mbits/sec, which is still under the proposed MPEG-1 upper bound of 1.85 Mbits/sec. The very first and last message additionally contain some header information. However, they encompass less frames. Accordingly a frame buffer of 250,000 bytes has been allocated.

With respect to the GOP pattern, a message usually is divided into six priority segments:

I	**9xB**	**P**	**9xB**	**P**	**9xB**

For sequence header information an additional segment might be added. Therefore the earlier proposed priority header consisting of 20 short integer values meets our requirements. However, the values of the priority table were slightly adjusted (Table 6).

Table 6: Red’s Nightmare: Priority distribution

Segment Type	**Priority**
Sequence delimiters	100
GOP header & I-frame	600
P-frames	750
Block of B-frames	900

4.5 Implementation Details

This subsection provides an basic overview over some details implemented in the simulation. It addresses the problem of coping with varying messages sizes for a given packet size and describes a simulation specific function that inserts dummy B-frames in the recovered bitstream. The latter feature is essential in order to display the original MPEG movie and the MPEG + PET sequence simultaneously.

4.5.1 Packetization Algorithm

Earlier in this chapter (Figure 15) we described the mapping of a group of pictures (GOP) onto packets of size 380 bytes each. This example did work exactly with the given values, but it is clear that for a varying GOP length and different priority distributions the segments not automatically fit into a given packet size. We solve this problem by introducing a simple packetization algorithm.

The following values are specified by the user:

- LM: Length of message
- PS: Packet size
- LM_i: Length of priority segment i
- %i: Priority of segment i

Then we can calculate a preliminary approximation for the total length of encoding as follows:

$$LE^{pre} = \sum_i LM_i / \%\mathrm{i}$$

Since it is only a preliminary value, the length of encoding is represented as a floating-point number. Knowing LE and the packet size PS we calculate the total number of packets needed:

$$NP^{pre} = LE^{pre} / \mathrm{PS}$$

Now for each segment i we determine the number of blocks with the corresponding priority:

$$NB_i = \left\lfloor LM_i / \left(\%\mathrm{i} \times NP^{pre} \right) \right\rfloor$$

Note that NB_i is truncated to the floor value since otherwise the priority of the segment could not be guaranteed. In case the block length $\left(\%\mathrm{i} \times NP^{pre} \right)$ is smaller than one, it has to be set equal to one.

The quotient of the segment length and NB_i denotes the size of each block:

$$block_i = \lfloor LM_i / NB_i \rfloor$$

It is very unlikely that for the first approximation the segment size could exactly be partitioned into chunks of the block size. Therefore PET usually has to align block sizes (Figure 19).

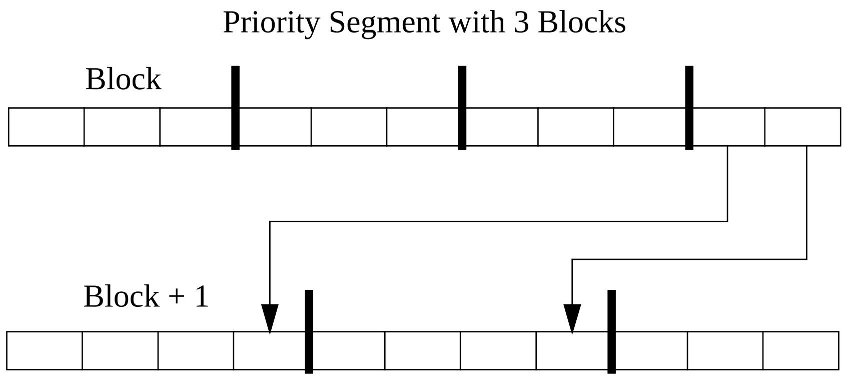

Figure 19: Block size alignment

Considering a sample segment consisting of 11 words (Figure 19) the number of blocks was calculated to three with block size three each. Consequently there are two words left which are added to the other blocks by increasing their size by one.

In general the number of expanded blocks NB2 is calculated as follows:

$$NB2_i = LM_i - (block_i \times NB_i)$$

$$NB1_i = NB_i - NB2_i$$

NB2 denotes the number of blocks with block size block+1 and NB1 refers to the original block size respectively.

By increasing the block length (degree of the polynomial) the segment does not match its priority anymore. Accordingly the first try for the total number of packets needed has to be updated:

$$NP_{max} = \max\ NP_i = \max \left\lceil \frac{(block_i + 1)}{\%\mathrm{i}} \right\rceil$$

Now we check whether we can reduce the redundancy by increasing the number of blocks within a segment. This is only possible in case there are some unused words in the packet left:

$$\sum_i NB_i < PS$$

Then we take the segment i which causes the highest amount of packets and increase the number of blocks by one. Thereby we decrease the size of at least some blocks. In case we manage to reduce all block sizes within a single segment, the number of required packets is consequently reduced as well. If the block size is already equal to one, we cannot lower the number of packets sent, but we recalculate the succeeding segments in order to exploit the total packet length by reentering in a previous processing step:

$$NB2_i = LM_i - (block_i \times (++NB_i))$$

$$NB1_i = NB_i - NB2_i$$

This loop is executed unless the size of the packet is exceeded. Thereby we do not reduce redundancy anymore, but the fault tolerance will be slightly increased. That is the effective priority of each segment is equal or higher than specified by the user.

4.5.2 Creating “Dummy” Bidirectional Frames

In order to keep the number of frames in the simulated bitstream the same as in the original MPEG file, a so called “dummy” frame replaces each lost frame. It is necessary to copy the last correctly decoded frame, which in this case could either be an intra or a predictive coded frame. Bidirectional frames consume the least amount of data and therefore were chosen as default frame type. With respect to the software architecture, M_retrieve() detects losses in the recovered data and then calls a function create_dummy_b() that builds up the frame.

In a MPEG-1 bitstream the sequence header contains pictures size information. Create_dummy_b() needs the horizontal and vertical number of pixels in order to allocate memory and to produce the corresponding amount of macroblocks. We also assume that a slice corresponds to a complete line in the image. Since a macroblock consists of 16x16 pixels, the above values can easily be calculated:

$$\text{number of slices in frame} = \frac{\text{vertical picture size (in pixels)}}{16}$$

$$\text{number of macroblocks in slice} = \frac{\text{horizontal picture size (in pixels)}}{16}$$

Initially the frame has to be marked as a bidirectional type on the picture layer, by inserting the picture_start_code, temporal_reference, picture_coding_type and frame buffer delay. They are followed on the slice layer by an increasing 32 bit slice_start_code, the quantizer_scale and an extra_bit_slice:

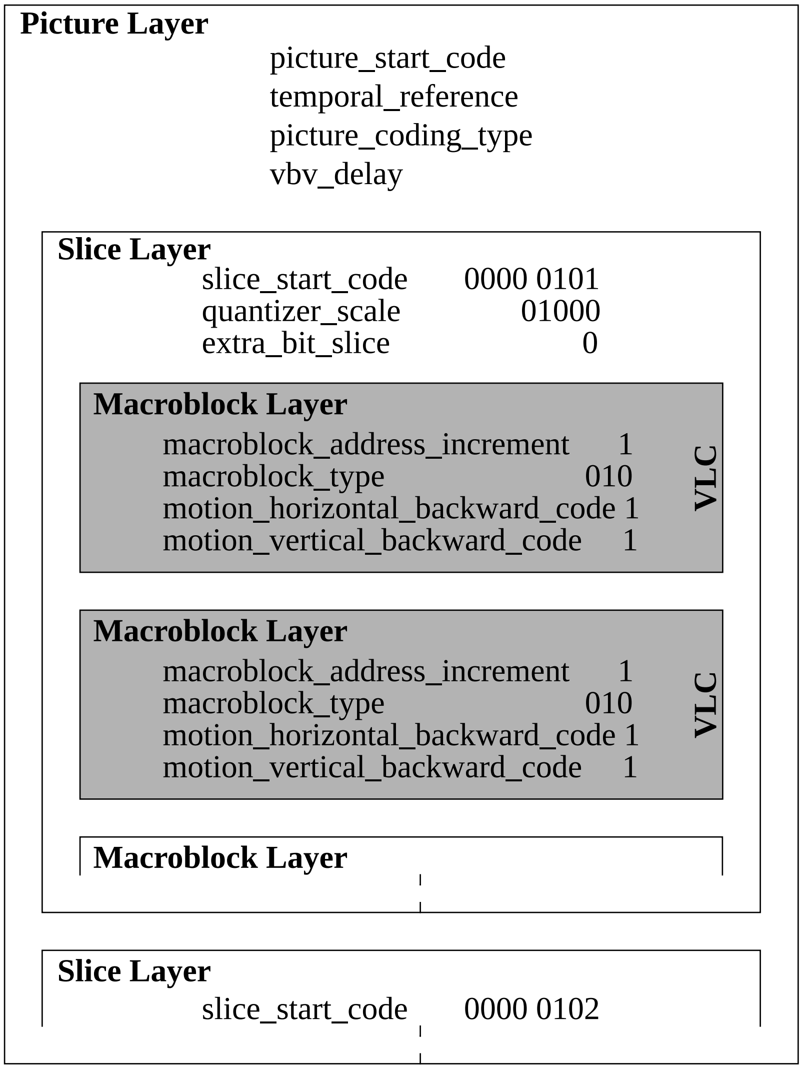
Picture Layer
picture_start_code
temporal_reference
picture_coding_type
vbv_delay
Slice Layer
slice_start_code 0000 0101
quantizer_scale 01000
extra_bit_slice 0
Macroblock Layer
macroblock_address_increment 1
macroblock_type 010
motion_horizontal_backward_code 1
motion_vertical_backward_code 1
VLC
Macroblock Layer
macroblock_address_increment 1
macroblock_type 010
motion_horizontal_backward_code 1
motion_vertical_backward_code 1
VLC
Macroblock Layer
Slice Layer
slice_start_code 0000 0102

Figure 20: Design of dummy B-frames

Note that on the macroblock layer values are variable length coded and therefore have to be looked up in special tables defined in the standard [ISO11172]. As macroblock_type the macroblock_motion_backward pattern was selected, since the goal was to copy the previous reference frame. Accordingly the motion codes are set to zero, since no motion vectors appear in the bitstream. These bit patterns are then byte aligned and repeatingly inserted until the whole image is created.

Note that “frame stuffing” does not work simply by duplicating predictive frames. They always refer to the previous reference frame which could be the P-frame itself. Consequently it would display the difference to itself which obviously does not lead to the desired results.

4.6 Statistical Analysis

We conducted a series of experiments in order to investigate the performance of our coding scheme. The simulation was tested on a SPARC 10 Quad (4 processors). We were able to run the three concurrent images at a speed of six frames per second. Consequently a message was updated every five seconds. In the previous chapter we introduced a MPEG sequence according to which we allocated resources. Statistics about a sequence with a different GOP pattern are presented later. From a network point of view we investigated the simulated loss patterns and discussed redundancy aspects.

4.6.1 Redundancy

Assuming a packet size of 2000 bytes plus 40 bytes priority header, we randomly select a message of the Red’s Nightmare simulation. Function Prio_calc() after four iterations returned the following values:

Table 7: Statistical analysis of sample message

Priority Segment	**I**	**9xB**	**P**	**9xB**	**P**	**9xB**
Number of blocks	206	170	121	194	122	187
Block length	27	41	34	41	34	41
Number of blocks with block length block +1	69	103	71	92	86	100
Number of blocks with block length block	137	67	50	102	36	87
Number of packets	47	47	47	47	47	47
Effective priority (x1000)	596	894	745	894	745	894

Note that the effective priorities match pretty well the values specified earlier in Table 6. Block specific values were calculated according to the packetization algorithm. The maximum degree of all polynomials in the sample message is 41, the length of the block. From a computational point of view this is an acceptable value. In general the computational effort depends linearly on the message to packet size ratio. However, in the sample sequence PET has to cope with polynomial degrees of up to 126. With respect to a coding/decoding software implementation this may add unacceptable latency. A new PET system based on Cauchy matrices promises to reduce significantly the computing time.

For the current message a total of 47 packets were encoded resulting in a total encoding length of 95,880 bytes. Regarding the original GOP size of 73,868 bytes this causes a redundancy of 29.8%.

Combining all 41 messages the movie is encoded into 2,280 packets. Table 8 compares the total encoding with other relevant figures:

Table 8: Comparison of overall redundancy

MPEG sequence	**Encoding**	**MJPEG equivalent**
3,619,896 bytes	4,651,200 bytes	15,180,660 bytes

The quotient of the total encoding and the original MPEG sequence determines the overall redundancy of 28.49%. Taking into account that it enables the I-frame to sustain losses of up to 40%, the P-frames of up to 25% and the B-frames of up to 10% of the packets, this should represent an acceptable overhead. The Motion JPEG equivalent still would be 3.26 times higher. It was calculated using the average I-frame size of Table 5.

The redundancy obviously is also dependent on the packet size. In the current version the priority header is transmitted with each and every packet. Hence it contributes more overhead in the case of small packets. On the other hand, if the packet size is too big, the priority distribution can not exactly be mapped onto the packets. For example the PET interface on the decoding side has to be able to recover sequence header information from any 10% of the received packets (Table 6). Therefore at least ten packets have to be sent which may result in an enormous overhead (the maximum protection is achieved by transmitting the total sequence header with each packet). As shown in Figure 21 our choice of packets is close to the minimum.

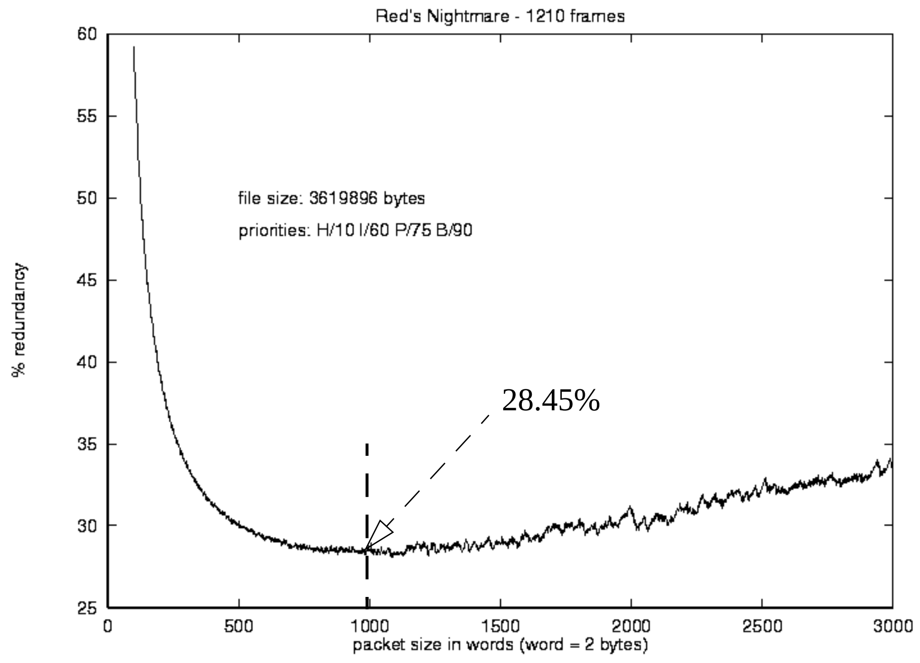

Figure 21: Redundancy vs. packet size

With respect to smaller packet sizes a different scenario proposes an encoded transmission of the priority table. An additional byte might be added representing the amount of packets needed in order to reconstruct the priority table. It is obvious that this extra priority has to be higher than the highest priority defined in the table.

4.6.2 Simulated Losses

As mentioned above we randomly threw away packets on a message basis in order to simulate a lossy medium. The algorithm was designed to simulate both uniformly distributed and burst errors. A peak loss of 41% packets lost per message has been detected. However, the average loss was 14%.

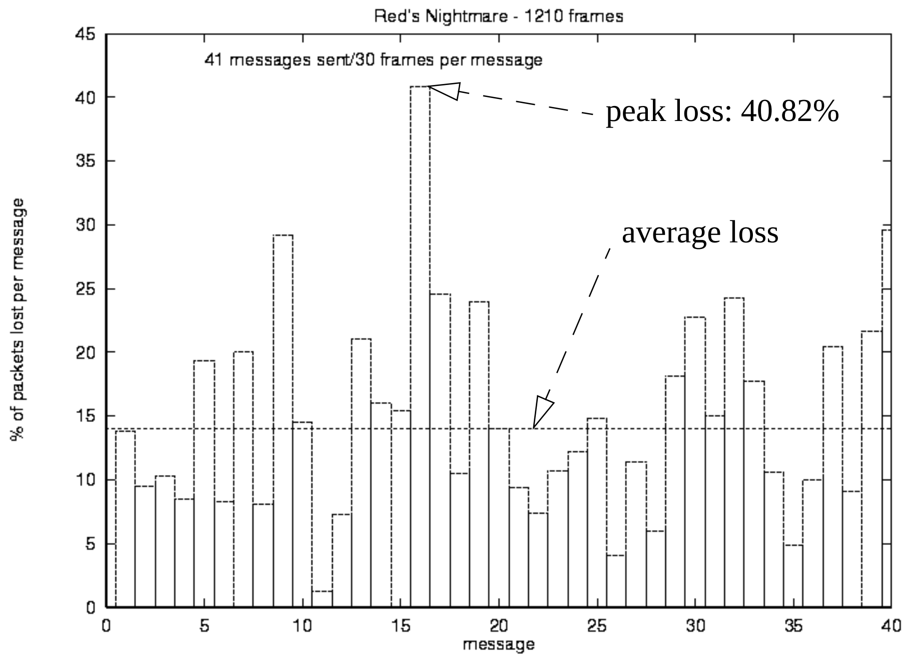

Figure 22: Red’s Nightmare: Packet loss per message

The presented loss pattern matches our design goals. It covers the total range from no disturbance (<10%) up to the loss of a total group of pictures (>40%). In the latter case the system has to make sure that the first B block of the succeeding GOP is replaced by dummy B-frames.

From the total amount of 2,280 packets sent we received 1,961. Figure 22 shows which frames have been affected by these losses. According to our redundancy distribution almost no reference pictures (I & P) have been lost, which was the main goal of the encoding process. The big gap of B-frames is a GOP pattern specific problem. In GOPs with less frames we would observe less frames missing. This is also the reason why the recovered movie tends to be a little jerky. Better results were obtained with a different motion picture.

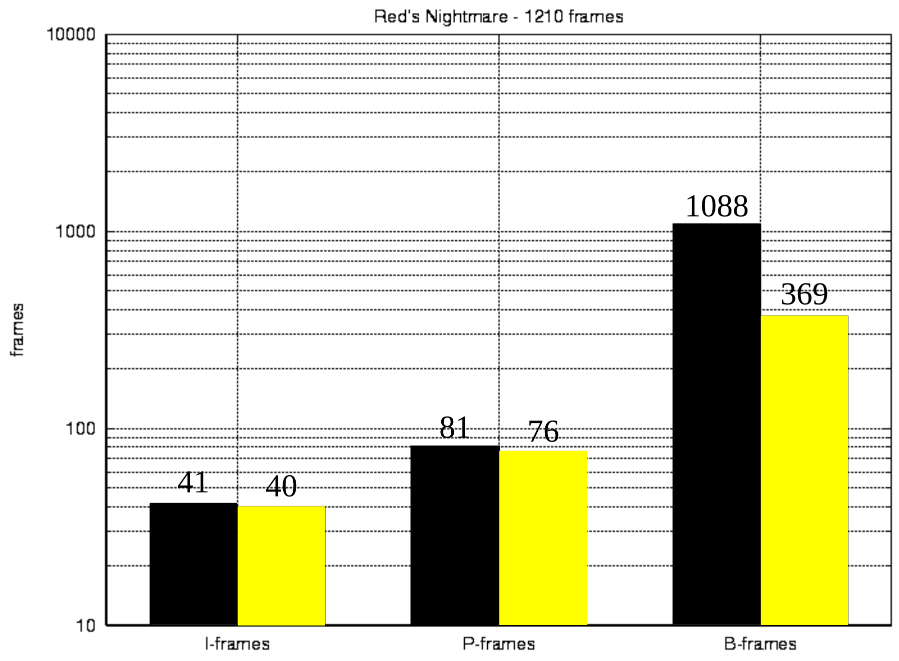

Figure 23: Losses affecting different frame types

The redundancy spent per message together with the simulated losses is shown in Figure 24. The figure displays three graphs versus the messages: The original MPEG file, the encoded bitstream and the received amount of data. Figure 25 displays the same data in terms of packets. It takes 142 packets to encode the biggest message. This corresponds to a total of 289,680 bytes or 2.21 Mbps. Note that since a GOP consists of 30 frames each the values directly correspond to the bandwidth consumed by the transmission. The difference between the MPEG and the encoding line represents the added redundancy.

You may observe that the line representing the received data crosses only three times the MPEG original. It would be the wrong conclusion to assume that consequently the original unencoded

MPEG bitstream would have been affected only three times by these losses. As shown in Figure 1, in the introduction, a bitstream without redundant protection would have been affected as well.

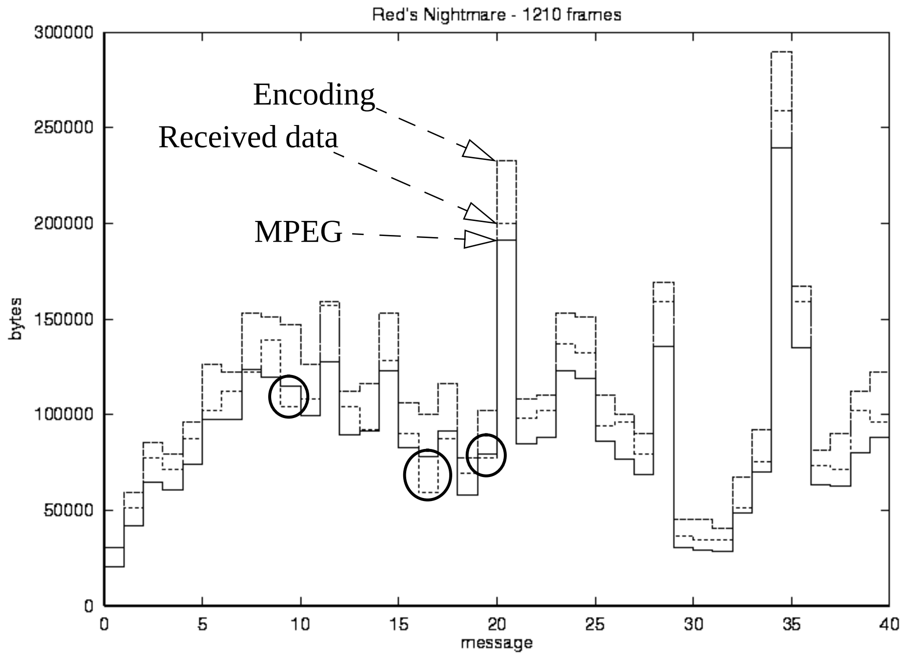

Figure 24: Red's Nightmare: Comparison of MPEG, encoding and received data

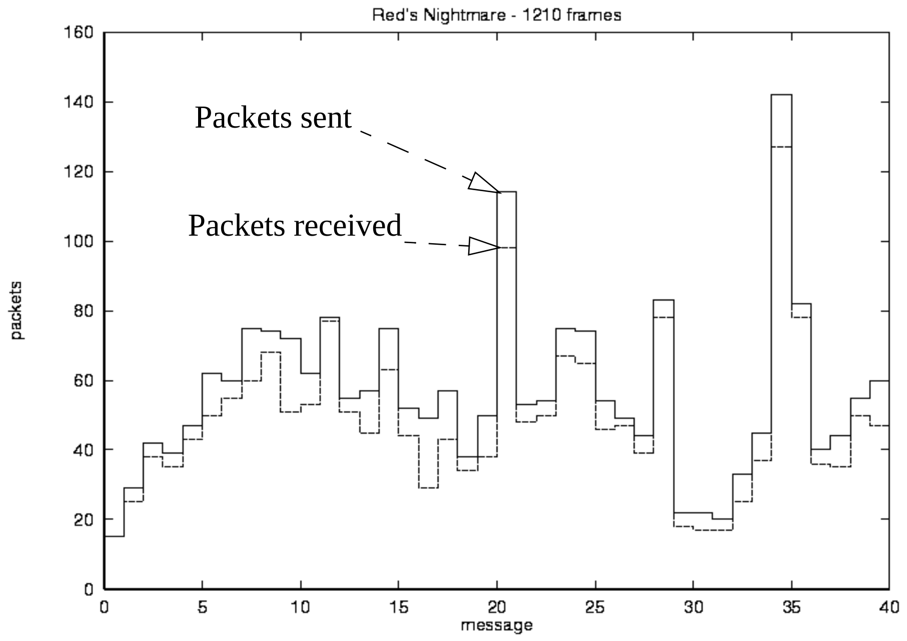

Figure 25: Red's Nightmare: Packets sent/received

For comparison, the manipulated sequence without PET protection shows MPEGs susceptibility to errors in the bitstream. Red's Nightmare is split into packets of size 2,000 bytes each. They were then randomly eliminated in the same way as in the PET demonstration. However, the packet loss probability was much less. Since we did not packetize them on a GOP basis we selected a window size of 50 packets. This value approximately corresponds to the average amount of packets sent per message in the PET case.

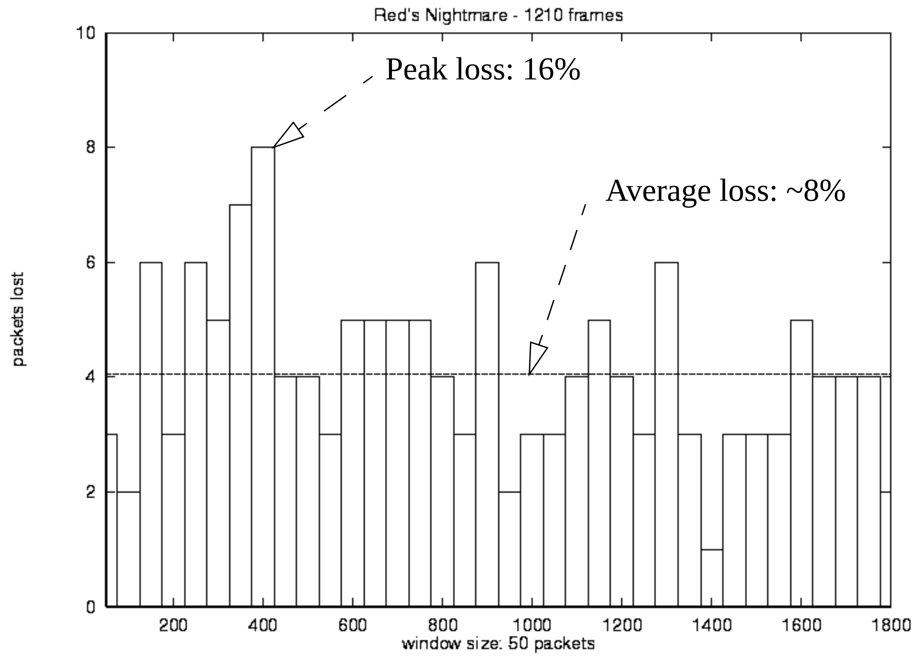

Figure 26: Simulated losses in Red's Nightmare without PET

In total, the sequence was partitioned into 1,810 packets of which 1,663 stayed in the remaining bitstream.

Even though the errors were small compared to the values of the PET encoded scenario, the picture quality is much worse. The Berkeley player even crashes completely after 38 frames. Whole lines of macroblocks are missing or display the difference to a disrupted reference frame. It has to be pointed out that a different loss pattern with the same average losses might disrupt the sequence even more depending on the amount of affected reference frames, while this would have no effect on the PET protected sequence.

4.6.3 A Different GOP Pattern

The choice of Red’s Nightmare for the current PET scheme has a certain drawback, since a low priority segment consists of 9 consecutive B-frames (Figure 18). As mentioned in chapter 4.4 27 frames have to be substituted whenever the losses per message exceed 10%.

We introduce a different sequence containing 750 frames of an ice hockey match. Its GOP pattern suits our purposes pretty well:

IBBPBBPBBPBBPBB

In this case even with losses of up to 25% at least every third frame is updated. The picture becomes less jerky and the amount of recovered information is significantly increased.

The Appendix contains related statistics about the hockey sequence. It also shows a comparison of both motion pictures.

5 Conclusion

We have presented a hierarchical encoding scheme based on Priority Encoding Transmission (PET). The feasibility of this scheme has been proven by introducing prioritized MPEG-1 bitstreams. Low quality video is achieved by receiving only reference frames, whereas quality is improved by the reception of more inter frames, representing the less critical data. Due to multilevel redundancy assignment they were primarily affected by simulated packet loss. It has been shown that no matter which subset of packets is extracted, the information can be recovered in priority order (Figure 23). Packetization according to a priority table has been described, causing a modest amount of redundancy compared to scenarios based on traditional Forward Error Correction (FEC).

Future research will be devoted to the enhancement of the current PET scheme. Apart from the idea described in chapter 2, faster software algorithms based on Cauchy matrices already exist which show promising results. On a SPARC 10, throughput rates of 3 Mbps and more have been achieved. In addition to speed aspects, the focus is on graceful degradation and latency. Unlike in the version used in our simulation, losses slightly higher than the redundancy specified by the application do not lead to the discarding of a whole segment. Consequently parts of the segment can still be recovered. A continuous PET version is also under development, which aims at reducing latency due to frame buffering. Smaller messages could also reduce computational complexity depending on the message to packet size ratio. This effort supports compatibility for ATM networks.

The search is open for the integration of other scalable applications. A video codec recently has been released by Taubman/Zakhor [TaZa93]. It possesses multi-resolution and multi-rate properties which could nicely be integrated in a PET scenario. Consequently, granularity would be shifted from a frame basis onto a picture quality layer.

While protocol suits improve channel allocation strategies and network performance, PET, integrated in gigabit testbeds, could enhance existing multimedia applications. Alternative ways of realizing the approach and a generic prototype are currently under development.

REFERENCES

[AlFe94] *Albanese A., Ferrari D.;* **Realtime Multiparty Multimedia Services and Applications for Collaboratory Experiments;** Internal paper International Computer Science Institut, September 1994

[AlBl94] *Albanese A., Bloemer J., Edmonds J., Luby M., Sudan M.;* **Priority Encoding Transmission;** 35th Annual Symposium on Foundations of Computer Science, IEEE Computer Science Press, 1994

[AlLu94] *Albanese A., Bloemer J., Edmonds J., Luby M.*; **Priority Encoding Transmission;** TR-94-039 International Computer Science Institute, Berkeley, CA, August 1994

[ChA94] *Chiang T., Anastassiou D.;* **Hierarchical Coding of Digital Television;** IEEE Communications Magazine vol. 32 pp 38-45, May 1994

[Ga93] *Garrett W.;* **Contributions Toward Real-Time services on Packet Switched Networks;** Technical Report CU/CTR/TR 340-93-20 Columbia University, New York, NY 10027, 1993

[Gall91] *Le Gall D.;* **MPEG: A video compression standard for multimedia applications;** in Communications of the ACM, 34(4):305-313, April 1991

[GoRo94] *Gong K., Rowe L.;* **Parallel MPEG-1 Video Encoding;** TR #811, Computer Science Division - EECS U.C. Berkeley, May 1994

[Ha93] *Hagenauer J.;* **Einfuehrung in die Codierungstheorie;** Script Lehrstuhl fuer Nachrichtentechnik Technische Universitaet Muenchen, SS 1993

[ISO11172] *CCITT Recommendation MPEG-1;* **Coded Representation of Picture, Audio and Multimedia/Hypermedia Information;** ISO/ IEC 11172 Geneve Switzerland, 1993

[ISO13818] *CCITT Recommendation MPEG-2*, **Coded Representation of Picture and Audio Information;** ISO/IEC 13818/Draft, Geneve Switzerland, November 1993

[LaEf93] *Lamparter B., Boehrer O., Effelsberg W., Turau V.;* **Adaptable Forward Error Correction for Multimedia Data Streams;** TR-93-009 University of Mannheim, Mannheim, 1993

[MP2/94] *MPEG Software Simulation Group;* **mpeg2encode/mpeg2decode/ mpeg2play;** Version 1.1, June 1994

[Pa92] *Pancha P., El Zarki M.;* **Prioritized Transmission of Variable Bit Rate MPEG Video;** in Proceedings of GLOBECOM ‘92, Orlando, FL December 1992

[PaZa92] *Pancha P., El Zarki M.;* **A look at the MPEG video coding standard for variable bit rate video transmission;** in Proceedings of INFOCOM ‘92, Florence, Italy, May 1992

[PaZa93] *Pancha P., El Zarki M.;* **Bandwidth Alocation Scemes for Variable Bit Rate MPEG Sources in ATM Networks;** in Proc. IEEE INFOCOM’93, vol 3, San Francisco, CA, 1993

[PaZa94] *Pancha P., El Zarki M.;* **MPEG Coding For Variable Bit Rate Video Transmission;** IEEE Communications Magazine p.54-66, May 1994

[Part94] *Partridge, C.;* **Gigabit Networking;** Addison-Wesley, Reading, Mass., 1994

[PaSm93] *Patel, K., Smith B., Rowe L.;* P**erformance of a Software Video Decoder**; in Proceedings of ACM Multimedia 93, Anaheim,CA 1993

[PeMi93] *Pennebacker W., Mitchell J.;* **JPEG: Still Image Data Compression Standard;** Van Nostrand Reinhold, New York 1993

[Row94] *Rowe, L.;* **Multimedia Computing;** ILP ‘94 Highlight Talk

[Sha92] *Shacham N.;* **Multipoint Communication by Hierarchically Encoded Data;** in Proceedings of IEEE INFOCOM ‘92, Florence, Italy, May 1992

[TaZa93] *Taubman D., Zakhor A.;* **Multi-Rate 3-D Subband Coding of Video;** IEEE Transactions on Image Processing**,** April 1993

APPENDIX

Table 9: Comparison of Red’s Nightmare and Ice Hockey sequence

	Red’s Nightmare	**Ice Hockey**
GOP pattern	IBBBBBBBBBPBBBBBBBBBPBBBBBBBBB	IBBPBBPBBPBBPBB
Sequence Size	3,619,896	5,015,093
picture size	320x240	352x240
packet size	2,000 + 40 for PET header	2,000 + 40 for PET header
total number of frames	1,210	750
avg. I frame size	12,546	12,543
avg. P frame size	8,086	10,411
avg. B frame size	2,252	4,593
total compression rate	1.30% of uncompressed 24 bit image	2.64% of uncompressed 24 bit image
messages sent	41	51
total encoding	4,651,200	6,803,400
transmitted redundancy	28.49%	35.66%
packets sent/received	2,280/1,961	3,335/2,963
I frames sent/recovered	41/40	51/50
P frames sent/recovered	81/76	200/192
B frames sent/recovered	1,088/369	499/230
Motion JPEG eqivalent	~15,180,660	~9,407,250

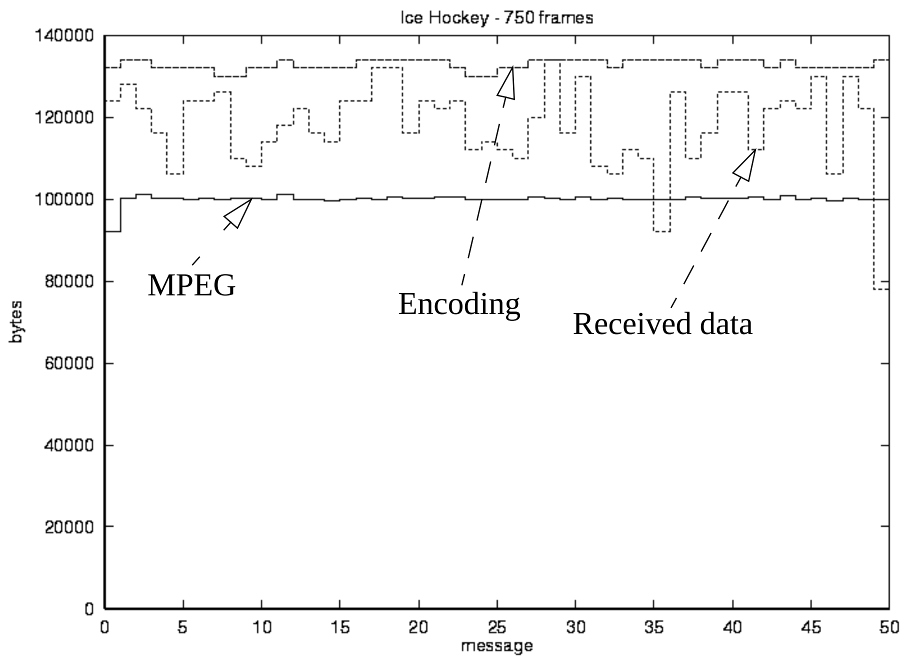

Figure 27: Ice Hockey: Comparison of MPEG, encoding and received data

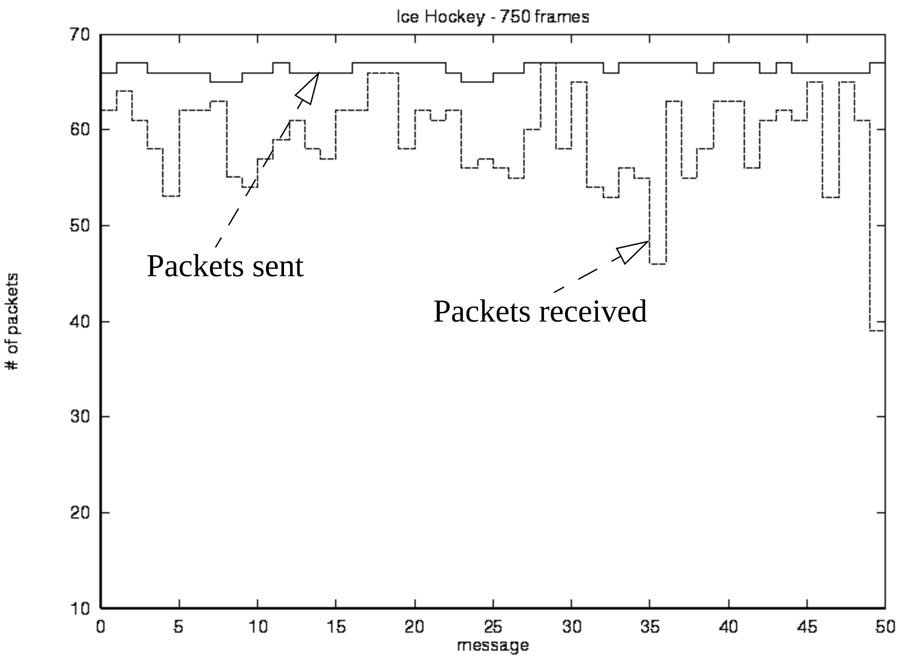

Figure 28: Ice Hockey: Packets sent/received

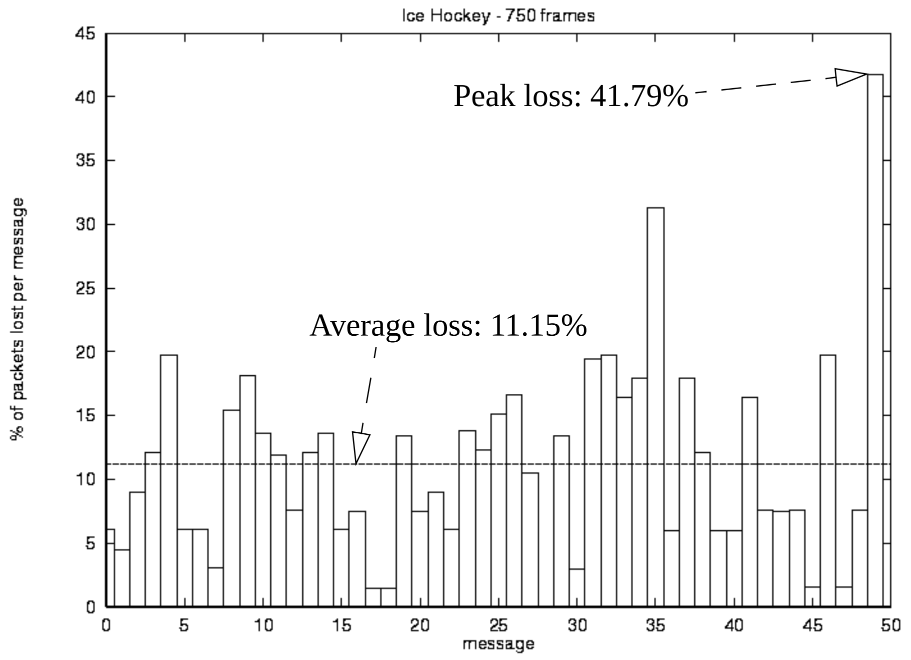

Figure 29: Ice Hockey: Packet loss per message

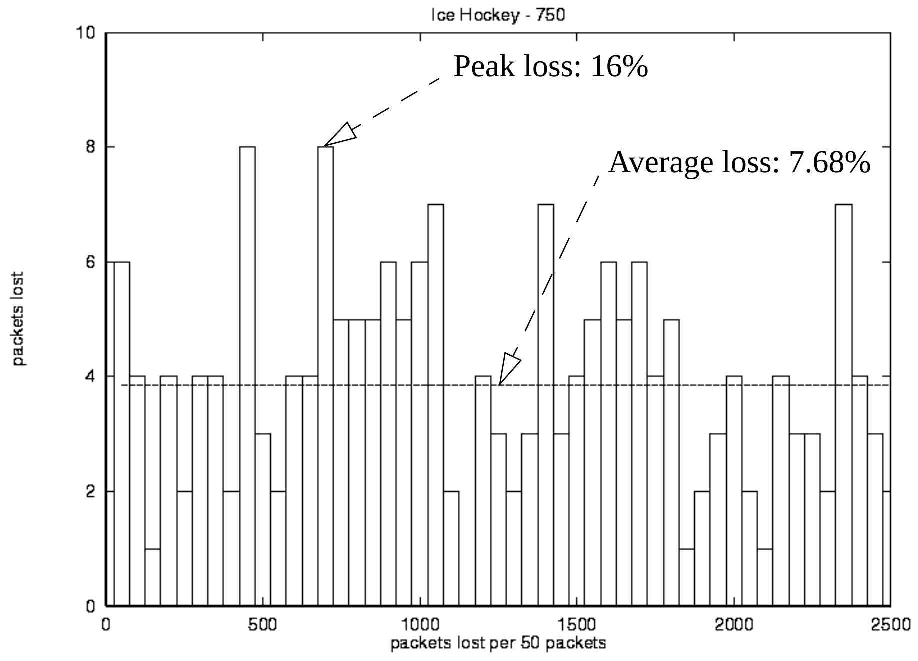

Figure 30: Simulated losses in Ice Hockey without PET