Robust Real-Time Videotransmission over Lossy Channels

Christoph Günzel* Falko Riemenschneider†

November 4, 1998

Abstract

We present a system for the transmission of high quality video streams such as MPEG-video streams in realtime over lossy channels. For protection against losses in the network during transmission we use a Forward Error Correcting (FEC) scheme called Erasure Resilient Transmission (ERT). ERT is an encoding scheme which provides multiple levels of redundancy in order to protect the different contents of a data set according to their importance. The task of assigning redundancies for the ERT encoding scheme is investigated for MPEG-1 encoded video sequences here. Furthermore we describe how to dynamically readjust the parameters of the ERT encoding scheme due to the changing sizes of the frames within a video sequence.

*Dept. of Computer Science, University of Bonn, 53117 Bonn. Email: guenzel@cs.bonn.edu

†Dept. of Computer Science, University of Bonn, 53117 Bonn. Email: riemensc@cs.bonn.edu

1 Introduction

The number of computer applications which make use of MPEG-1 encoded video sequences is continously on the rise. At the same time many of these applications are intended for usage in a heterogeneous networking environment which exhibits links with different bandwidths and workstations with varying computational power.

In many communication situations, data is lost in transit. A standard response to this problem is to request retransmission of data that is not received. The network transport protocol TCP [St 94] deals with that error correction scheme. When some of this retransmission is lost, another request is made and so on. Such communication protocols like TCP are often the source of delays in networks and are a disadvantage for realtime applications. To avoid these delays we use the UDP [St 94] protocol. The UDP protocol does not have an error correction mechanism.

Therefore we have to protect our messages against losses in the network during transmission. Here we use a mechanism called Error Resilient Transmission (ERT) related to [BKK$^+$ 95] [G 96] [ABEL 94]. ERT is a Forward Error Correction scheme with several priority levels. With this method one can initially transmit extra redundant information along with the raw message so that the message can be recovered from any sufficiently large fraction of the transmission. A major goal of ERT is to provide graceful degradation for MPEG-1 encoded video sequences in order to allow users with different hardware equipment and connectivity to share the same application. ERT protects the different contents of MPEG-1 video according to their importance via a multilevel redundancy scheme and information spreading [Le 94] [GRW 97].

The intention is to increase the likelihood that the more important parts of the information can still be recovered if the video sequence is corrupted by packet losses during transit. These packet losses can be caused by different events like network congestion, fading in satellite transmission, insufficent computational power of the receiving workstations and so on.

Coding the video streams in such a way that a potential degradation is perceived as graceful is a challenging task. First it must be clear, how the different quality reductions in a video sequence are perceived by the human eye. These reductions affect basically colour, spatial and temporal behaviour of the video as well as overall precision of the pixels.

A thorough understanding of the MPEG-1 as well as the ERT encoding process is then necessary to analyze what kind of changes lead to which effect, assuming that a certain packet loss behaviour is given.

In chapter 2 we give an introduction to several networking protocols. In chapter 3 we give basic information about the MPEG-1 standard. In chapter 4 we describe and discuss the ERT encoding scheme. In chapter 5 we introduce the resulting ERT encoding scheme for transferring MPEG-1 video sequences and analyze the changes of quality due to packet losses. Unfortunately, the sizes of the frames within a video sequence differ depending on types and scene content. With ERT, it is possible to adjust the parameters very easily for encoding, decoding and transmitting.

2 A protocol to transmit data via UDP/IP

The Internet Protocol (IP) is a packet oriented transmission service for heterogenous networks. Every machine connected to the internet has its unique IP address. In fact the internet is a compound structure of many different networks. Routers and bridges connect the sub- and super-networks. If a machine crashes or a cable burns the routers themselves choose other routes for packet delivery (if possible). However this packet oriented delivery service for heterogenous networks may loose or destroy packets. In general those failures are called transmission errors. These errors may be:

1. Change of packet content.
2. Change of order among two or more packets.
3. Duplication of packets.
4. Packet loss.

Error type 1 is no problem for any protocol based on IP since IP guarantees that if a packet reaches its destination its content is correct. Error type 2 and 3 are easy to detect and the correction is trivial. By identifying every packet with a monotonous increasing number we can reorder packets or throw duplicates away. This number has another advantage: packet loss (error type 4) is being detected immediately. In case of packet losses (error type 4) we may either request retransmission of lost data and risk delays or try to perform without the lost piece of data.

Applications using the internet have the choice among two basic protocols for data transmission.

The Transmission Control Protocol (TCP/IP) is a connection oriented protocol. Before transmitting the data an application has to establish a logical connection between itself and the remote host. The end points of a connection are called sockets. A socket is uniquely identified by the IP address of the network interface card and a port number.

Thus every connection is uniquely identified by a pair of sockets. Sockets used for TCP connections are called stream sockets. TCP controls the flow of packets. The connection is safe in the way that logically no data will be lost. If packets are lost the sender misses the receivers acknowledgement and starts a retransmission. All types of transmission errors will be corrected fully transparent for the application level. For most applications TCP is the right choice. Many errors lead to much waiting but users are used to it.

On the other side there is the User Datagram Protocol (UDP/IP) which uses the bare IP service of transmitting packets. Except error type 1 (the change of packet content) the application level itself has to deal with all types of transmission errors. Passing data through the network using UDP is called connectionless. The sender creates its socket and sends the data with explicit specification of the IP address and the port number. If the destination machine doesn't exist or there is no socket bound to the specified port or the receiving process isn't ready to receive the packet it is lost. The sender receives no acknowledgement of any kind. Therefore this protocol has less overhead than TCP. The transmission quality depends on

- Network quality
  - Load of routers
  - Routing strategies
  - Hardware failures
- Load of receiver machine
- OS specific implementation of IP

Since we can't afford any delay for our application, the videotransmission, we choose UDP for our purpose. Here we introduce a simple flow control scheme for sending massive amounts of data via UDP. The forces are:

- The correctable types of transmission errors affecting UDP applications (error 2 and 3) shall be corrected.
- Without any kind of synchronization numerous packets will be lost because the receiver isn't ready to process incoming UDP datagrams. But a handshake to synchronize sender and receiver costs time so handshaking will take place only if it is necessary.
- The sender may be much faster than the receiver so again we loose packets due to buffer overflows in the receiver's machine. Our flow control protocol will slow down the sender so the receiver may catch up. This delay depends on the receivers performance. If the receiver is as fast as the sender no delay shall occur.

- In order to track the amount of packet losses our protocol should exchange meta information.
- We need the notion of a *job of packets* to control when handshaking may occur and to calculate the loss percentage on the basis of a discrete number of packets.

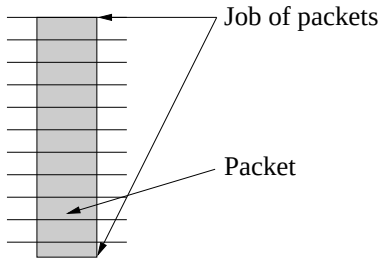

Figure 1: A block of data as a job of packets

A job is a sequence of n packets each of size s. Let D be the amount of data to be sent in bytes, then $n = \lceil D/s \rceil$ (see figure 1). Each packet of a job contains the id of it's job and the packet id. Identificators are unique, monotone ascending numbers (see figure 3).

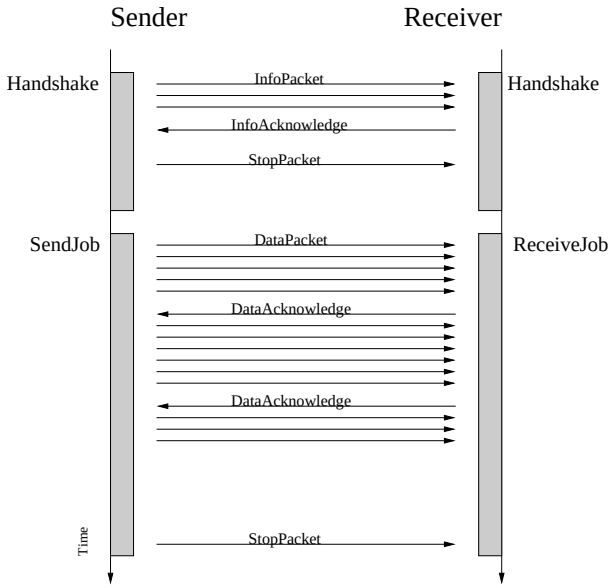

Figure 2: Overview of the protocol

Handshakes can only occur right before or after a job of packets. The sender will initiate the synchronization procedure if it is necessary. The sender pushes it's packets into the network while concurrently listening for acknowledgements. These will not be sent back for every packet the receiver processes but on a regular basis given by a certain period of time. If the sender misses these packets for too long he times out and stops the transmission.

Data (25+nSize bytes)		
Offset	Type	Description
00	char	szVersion[8] - version
08	byte	nType = PACKET_DATA
09	ulong	nJobId - job id
13	ulong	nTotalJobSize - job size in bytes
17	ulong	nPacketsSent - number of sent packets
21	ulong	nSize - packet size
25	char	pCharArray[nSize] - data to be sent

Data Acknowledgement (33 bytes)		
Offset	Type	Description
00	char	szVersion[8] - version
08	byte	nType = PACKET_DATAACK
09	ulong	nJobId - job id
13	ulong	nPacketsReceived - number of received packets
17	ulong	nLastPacket - highest packet id received
21	ulong	nBytes - bytes processed
25	ulong	nTimeInside - msecs used to received
29	ulong	nTimeOutside - msecs used outside receiver

Figure 3: Packet formats for data transmission

The acknowledgements (fig 3) help the sender in three different ways:

1. Determining when a handshake is necessary.
2. Calculation of the average time period the receiver needs to process one packet
3. Generating information about packet loss

When is a handshake necessary? Let j_s be the packet id of the last sent packet and j_r be the packet id of the last packet received. We call $j_s - j_r$ the senders lead. If this lead exceeds a given threshold we assume that a synchronization is necessary to avoid massive packet losses.

How long should the sender delay take between two data packets? The sender measures the time it needs to send a packet. Since the receivers acknowledgements inform

the sender about the number of received and processed packets the sender can easiliy calculate the delay of the next packet. This calculation has a flaw because packets lost by the network will increase the senders delay allthough the receivers performance may be sufficient. Nevertheless it is likely that a router threw the missing packets away due to a buffer overflow. Waiting longer before sending the next packet will decrease the routers load so even in this case the adjustment is ok.

How can the loss rate be measured? This is a trivial calculation because every acknowledgement informs the sender about the number of received packets.

Summation of our achievements:

- With a good network quality very few packets are lost.
- Errors of type 2 and 3 are corrected transparently.
- Higher application levels can gain detailed loss information.
- Little extra delay due to limitation of the number of handshakes.

We must admit an overhead due to synchronization and packet headers. This overhead is substantial for packet sizes less than 100 bytes.

3 MPEG-1 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 bitrates less than 1.856 Mbits/sec. The MPEG encoding algorithm was developed primarily for storage of compressed video on digital storage media. Provisions were therefore made in the algorithm to enable random access, fast forward/reverse searches, and other features when decoding from any digital storage media. Image compression usually is achieved by applying several techniques such as subsampling of chrominance components, quantization, frequency transformation by Discrete Cosine Transform (DCT) and variable length coding (VLC) [PM 93] [Ga 91].MPEG additionally introduces motion compensation (MC) to exploit temporal redundancy, predictive coding and picture interpolation.

3.1 Coding Layers

The MPEG model can be organized into four layers as illustrated in Figure 4. The layers are arranged below in order of increasing size:

- **Block**- A block is the smallest coding unit in the MPEG algorithm. It is made up of 8×8 pixels and can be one of three types; luminance (Y), red chrominance (Cr)

and blue chrominance (Cb). The block is the basic unit in intraframe DCT coded frames.

- **Macroblock**- A macroblock is the basic coding unit in the MPEG algorithm. A macroblock is a 16×16 pixel segment in a frame. Since each chrominance component has one-half the vertical and horizontal resolution of the luminance component, a macroblock consists of 4 Y, 1 Cr and 1 Cb block.
- **Slice**- A slice, which is a horizontal strip within a frame, is the basic processing unit in the MPEG coding scheme. Coding operations in blocks and macroblocks can only be performed when all pixels for a slice are available. A slice is an autonomous unit since coding a slice is done independently from its neighbors.
- **Picture**- A picture in MPEG terminology is the basic unit of display and corresponds to a single frame in a video sequence. The spatial dimensions of a frame are variable and are determined by the requirements of an application.

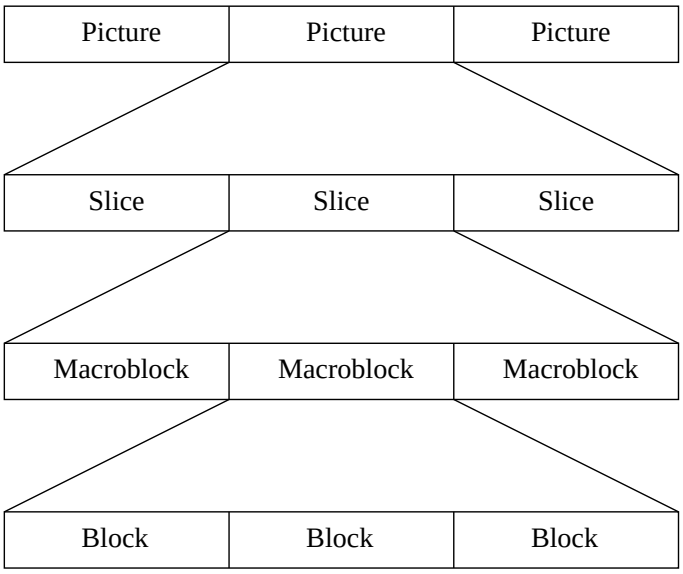

Figure 4: MPEG coding layers

3.2 Coding Modes

MPEG uses three different types of pictures: Intraframes (I), coded as a still image, Predicted frames (P), predicted from the most recently decoded I- or P-frame and Bidi-

rectional 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 [PM 93] [Wa 91]: Blocks are first transformed from the spatial domain into the frequency domain using DCT. The corresponding coefficients then are quantized in frequency order. 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. 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. Bidirectional frames therefore provide the highest amount of compensation [CRM 93].

B-frames are not used for prediction of other frames so the errors in them do not propagate, as opposed to the errors in I- or P-frames. This makes the B-frames the least critical of the three frame types. Errors in I- or P-frames propagate until the end of the GOP.

3.3 Bitstream Hierarchy

An MPEG-1 video is represented by a layered bitstream. The top layer is called sequence which is encapsulated by the sequence delimiters header and trailer. The sequence header contains information such as picture size, picture rate and bit rate. The trailer consists of a 32 bit 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 decoded independently.

A GOP consists of a GOP header beeing followed by different types of frames. In every GOP there is one I-frame which immediately follows the GOP header if the GOP is viewed in bitstream order. (further details concerning bitstream and display order are

described in [Le 94], [CA 94])) The I-frame may be followed by any number of I-, P- and B-frames in any order. The smallest GOP might consist of only a single I-frame, whereas the upper bound is unlimited. A GOP always begins with a GOP-header and either ends at the next GOP-header or at the end of the sequence.

An example for a GOP is displayed in figure 5.

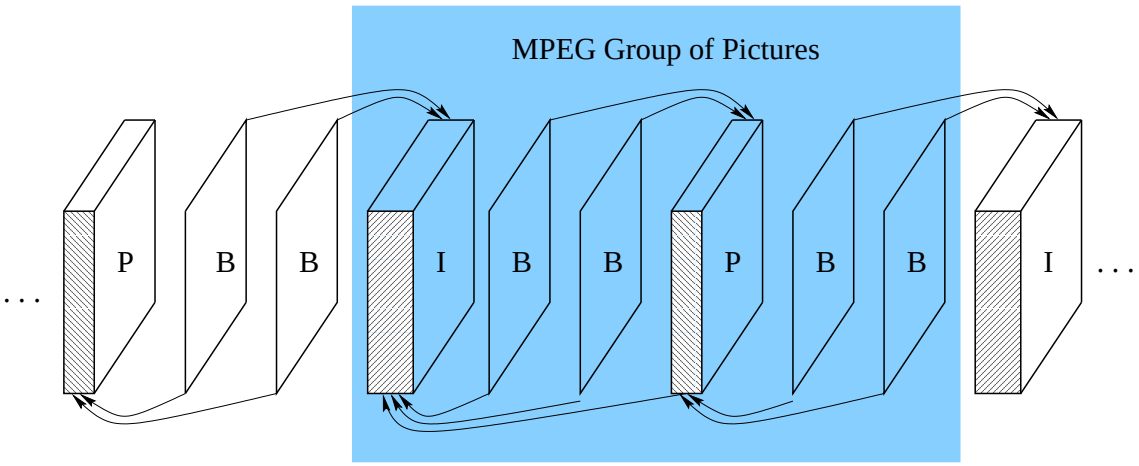

Figure 5: MPEG-1 Video Stream

There is another important point to mention considering GOPs. There is no such thing as a fixed size for all three frame types. The I-frame size can vary in the different GOPs of a video sequence depending on the scene content. The sizes of P- and B-frames can vary even within a GOP. That means that we have to readjust the parameters of the encoding scheme dynamically with the changing of the sizes of the GOPs and fames within a video sequence.

4 Erasure Resilient Transmission (ERT)

From the discussion in chapter 3 it has become clear that due to the different error propagation properties I-, P- and B-frames differ in their importance concerning the information content of a GOP. The main idea in ERT is to assign a certain amount of redundancy to a GOP with the redundancy being unevenly distributed among the different frame types, according to their importance [Le 94]. ERT is at the moment implemented based on encoding of information via Cauchy matrices [BKK$^+$ 95] [G 96]. The basics of the ERT encoding scheme can be explained with the aid of figures 6 and 8.

ERT encodes the data using multilevel redundancy and disperses the encoding into the packets to be transmitted.

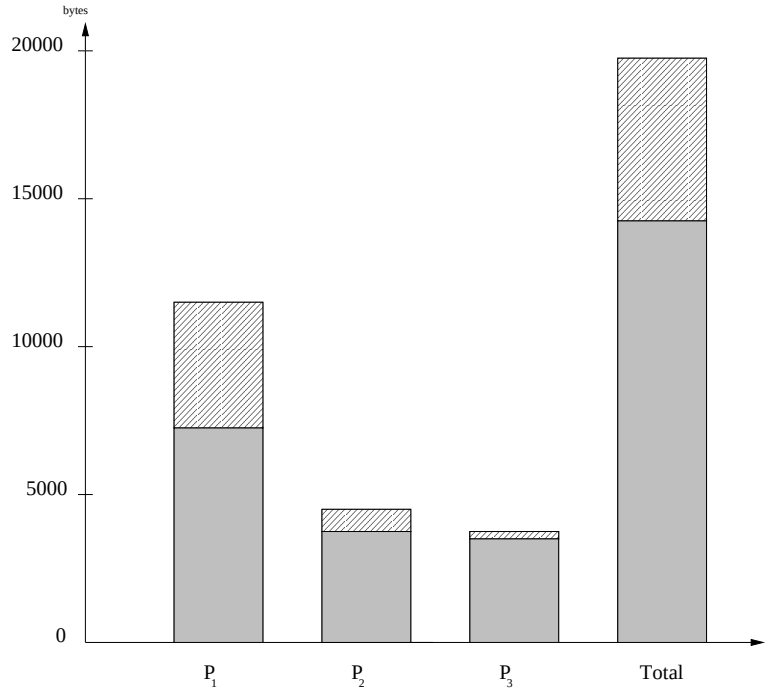

Figure 6: Multilevel redundancy distribution

In more detail: Given a message $M = (M_1, M_2, \ldots, M_m)$, -the message M is split in m packets for transmission- the ERT encoder generates a Code $C = (C_1, C_2, \ldots, C_n)$ containing n packets. The first m packets of C are the m packets of the message M, the other $n - m$ packets are the redundant information for the message. The ERT coding scheme guarantees, that for any sufficiently large fraction of the transmission, the receiver is able to recover the information. Therefore the ERT decoder is able to recover Code C if $\binom{n}{m}$ packets arrive at the receiver. For different parameters n and m, C contains different amounts of redundancy for the different priorities. Figure 7 shows a possible redundancy distribution for a typical MPEG-GOP frame pattern: I B B P B B.

Priorities are expressed by fraction of packets needed to recover the original information. According to the considerations in figure 7, 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.

Priority level	Size in Bytes	Occurrence in group of samples	Total Size	Fraction x_i needed to recover	Encoding
I-frame	12000	1	12000	60%	20000
P-frame	4800	1	4800	80%	6000
B-frame	2850	4	11400	95%	12000
Total		6	28200		38000

Figure 7: Possible redundancy distribution

In Figure 8 the generation of the code is explained very clearly for MPEG video streams. It can be seen that a GOP, which constitutes the message to be encoded, is encoded in a code C consisting of n packets. The mapping onto the n packets is done in such a way that information from every frame is contained in each of the n packets. As a consequence the information is spread among the n packets which renders improved robustness in the presence of bursty errors which are common in today's networking environment. The second idea in ERT is to provide error correcting properties on a multilevel basis. The most important data, the I-frame is endowed with relatively more redundancy information than the less important P- and B-frames (see also figure 6. Another property of the ERT encoding scheme is the fact that on the decoding side no decoding is required if the cleartext information arrives undisturbed. Furthermore, the encoding and decoding algorithms are dynamic that means, that in case the frames are different in size which is usual, the encoding and decoding algorithms choice their parameters due to the sizes of the message. The transmission of the encoded message is described in more detail in the following chapter 5.

In case of errors the amount of redundancy being assigned to the different frame types decides whether the frames of the specific type can be recovered or not. If enough error free packets arrive, all of the frames belonging to a certain frame type can be recovered. If this threshold is not reached, no frame recovery is possible via decoding. Nevertheless, some cleartext information might have gotten through so that there is still a chance that some usable information has arrived, even though the recovery mechanism does not have enough packets to recover the whole message.

The ERT encoding scheme renders a usable degradation of quality in cases of errors. Figures 9 and 10 show the difference between a singlelevel and multilevel redundancy distribution. Recall that x_j,j=I,P,B is the fraction of packets needed to recover frame type j. In the singlelevel case there is only one value $x = x_I = x_P = x_B$ which determines

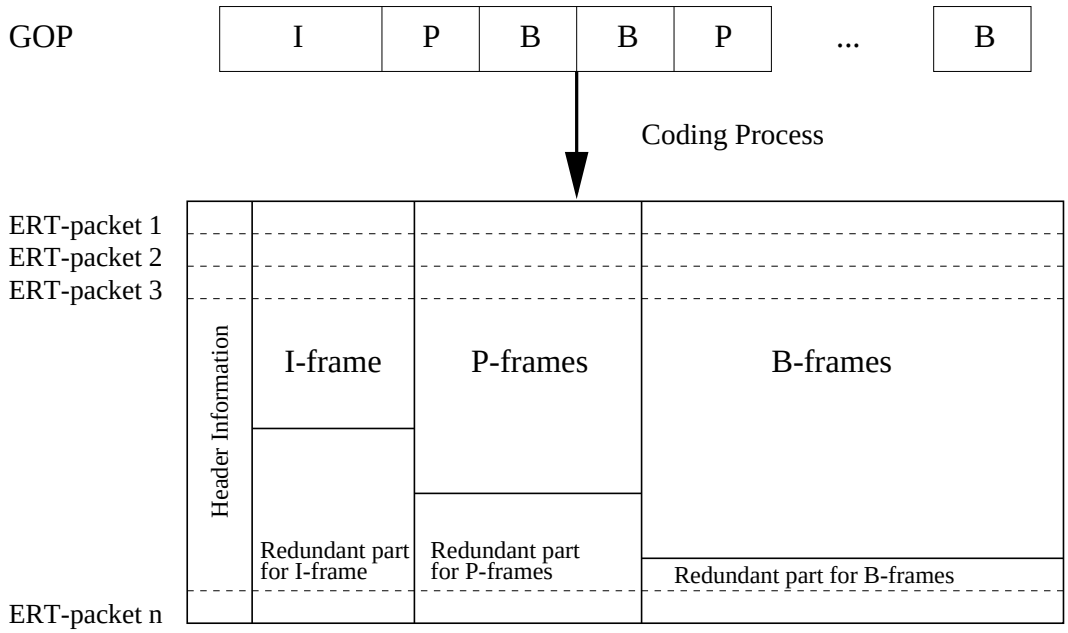

Figure 8: The coding process in ERT

the recovery threshold. Thus $1 - x$ is the fraction of packets allowed to be lost. When more than $1 - x$ packets are lost the quality of the video will be unacceptable. In the other case the quality will be good. In the multilevel case we have three thresholds x_I, x_P and x_B. The ERT encoding scheme generates a transition band between good and bad quality with less chance of bad quality. A disadvantage is that the threshold where the good quality starts is lower compared to the singlelevel case. But the generated redundancy overhead will be smaller and thus less bandwidth is needed.

5 Real-Time Transmission

In this section we present the combination of our simple MPEG GOP parser, the ERT codec and the UDP flow control protocol. The result is a C++ implementation running on an Ultra Sparc 5.

We developed a simple tool which traces UDP packet transmission based on our flow control protocol. We introduce the notion of the transmission context, which consists of

- a route used for packet delivery,
- a pair of machines (used as sender respectively receiver),

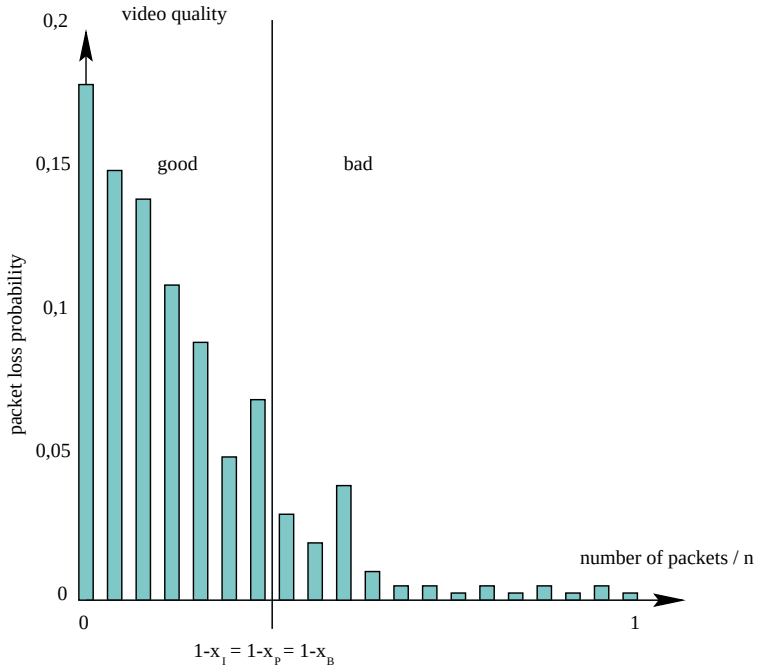

Figure 9: ERT with singlelevel redundancy

- a period of time in which transmission took place.

Without formalization our tests lead to the following assumptions concerning UDP data transmission quality:

- The UDP packet size does matter. Some contexts show a good performance using small packets (below 512 bytes) while producing high loss rates using high packet sizes (above 4 or 5 kb). Other contexts deal bad with small packets.
- The job length does matter. Remember that a job consists of a sequence of equally sized UDP packets. Using long jobs may reduce the likelihood that handshakes occur often. On the other side due to long jobs we may produce heavy packet loss because sender and receiver are not synchronized any more. It depends on the context.

Therefore let us assume that the UDP packet size S and the maximum job length P is given for a certain context. In practice we may run a simple tracing tool which tests a given context for different parameters, determining a packet size and job length which performs well in this context. Furthermore let us consider the percentage of redundancy added to each frame as given for a certain application and context. The redundancy ratio

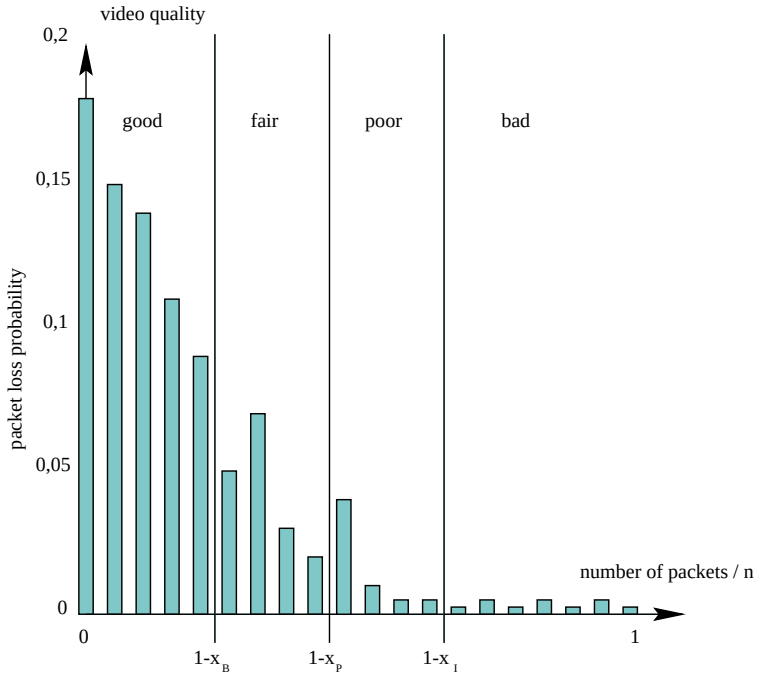

Figure 10: ERT with multilevel redundancy

is set for each priority level (R_h, R_I, R_P, R_B). These values can be easily calculated from the sufficient fraction of successfully received data which is needed to restore the whole frame.

Recall that the ERT encoder processing is based on a message M consisting of m packets. The encoder generates a code C consisting of n packets. Thus $n - m$ is the amount of packets containing redundant information. The first m packets in C are the original packets taken from M. Now our task is to map the MPEG frame data into packets for the ERT encoder.

For now on we will handle header information as frame data to maintain consistency in notation and implementation. Let N denote the number of frames F_i $(1 \leq i \leq N)$ in a GOP. T maps a frame F_i to its type $\in \{h, I, P, B\}$. $L(F_i)$ denotes the length of a frame measured in bytes. Then $m_i = \lceil L(F_i)/S \rceil$ is the number of original packets for a frame F_i. The ERT encoder will produce

$$n_i = \lceil m_i \cdot (1 + R_{T(F_i)}) \rceil$$

packets for each frame. The number of packets created for the whole GOP with N frames

(including header information) is therefore

$$n = \sum_{i=1}^{N} n_i.$$

See figure 11 for an illustration of the calculations above.

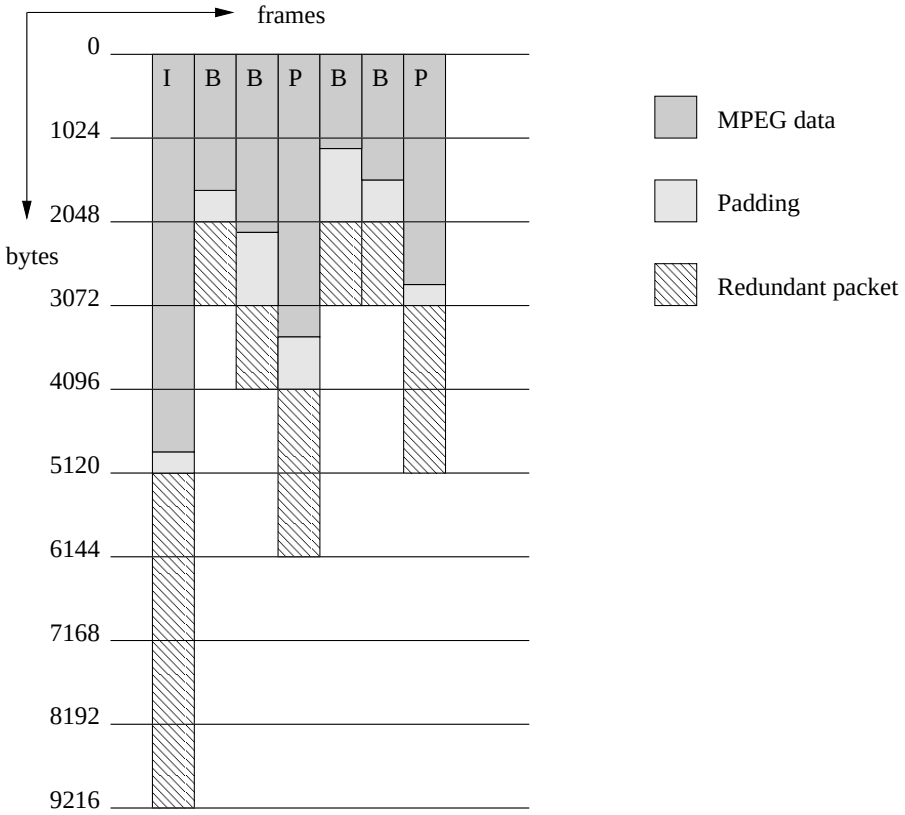

Figure 11: Encoding a GOP using the UDP packet size $S = 1024$.

Figure 12 shows the architecture we have chosen for our implementation. The raw video bit stream is produced by an MPEG encoder. The GOPs are parsed by a simple algorithm which scans in linear time for MPEG startcodes. The result is an ordered set of MPEG-encoded frames. Every frame has a type (that is: header, I-frame, P-frame or B-frame). The data is not modified, we only read and split one GOP into single frames.

After encoding the MPEG frame data we have n packets. We apply the given maximum job length P so $J = \lceil n/P \rceil$ is the number of UDP jobs we have to transmit. Our task is now to distribute the different packets by their type T over J jobs.

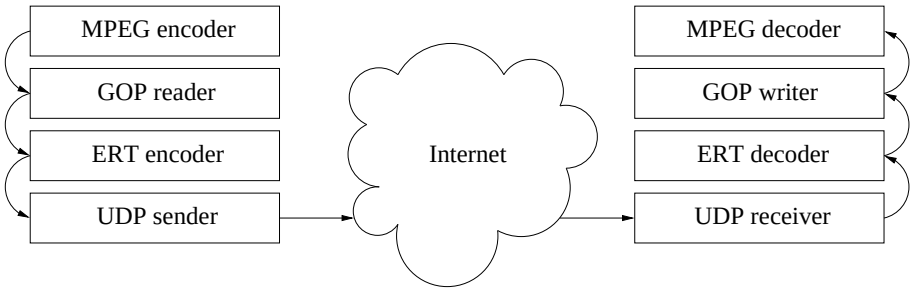

Figure 12: The building blocks of our architecture.

Each frame F_i consists of n_i packets (redundant or original) which have to be sent. We use J jobs so in average a frame F_i occupies n_i/J packets per job. For every frame we use a floating point counter f_i and round(f_i) denotes the number of packets a frame actually occupies in the next job. Before the first job begins we have $f_i = n_i/J$. The following job will consist of round(f_i) packets of frame F_i. In average $\sum_{i=1}^{N}$ round(f_i) will not exceed the maximum number of packets per job P. After a job is sent we set

$$f_i = f_i - \text{round}(f_i) + n_i/J.$$

The following example (figure 13) uses the GOP from figure 11. It assumes $P = 10$ which leads to $J = 4$. This GOP consists of $N = 7$ frames and $n = 33$ packets each of size $S = 1024$.

Before sending encoded data, sender and receiver synchronize themselves and exchange vital parameters for the ERT codec and information about the encoded GOP (number of encoded packets, number of jobs etc.) The UDP sender may then start the first job for that GOP and the receiver stores all received packets sorted by frame using the packet-id. Due to the fact that our protocol stores the job-id in every UDP packet the receiver can detect when the GOP is finished. It then calculates which frame reached it's destination with a sufficient number of packets. Those that can not be recovered are thrown away or replaced by dummy frames to maintain the video rate. Those frames that can be decoded successfully are piped directly to an MPEG decoder or MPEG video player.

Our scheme has approven to increase the quality of video transmission significantly. Although it remains impossible to transmit high quality video near real-time using low bandwith IP connections, no matter what scheme one applies, our architecture reaches our goals.

Frame i	1	2	3	4	5	6	7	
Type $T(F_i)$	I	B	B	P	B	B	P	
$f_i = n_i/J$	2.25	0.75	1	1.5	0.75	0.75	1.25	1.job
round(f_i)	2	1	1	2	1	1	1	9 packets
f_i after 1. job	2.5	0.5	1	1	0.5	0.5	1.5	2.job
round(f_i)	3	1	1	1	1	1	2	10 packets
f_i after 2. job	1.75	0.25	1	1.5	0.25	0.25	0.75	3.job
round(f_i)	2	0	1	2	0	0	1	6 packets
f_i after 3. job	2	1	1	1	1	1	1	4.job
round(f_i)	2	1	1	1	1	1	1	8 packets
Summary	9	3	4	6	3	3	5	33 packets

Figure 13: Distributing packets over jobs

Figures 15 and 14 show a movie scene from the video Foreman. On the left the original is displayed, in the middle the video with ERT protection and on the right the same sequence without ERT protection.

6 Conclusion and further research

We have presented a useful method to transmit medium quality video sequences such as MPEG-1 to different users across a lossy network. We have shown how to prioritize the MPEG-1 video sequences in a way that a graceful degradation of quality is possible in case of packet losses caused by the networking environment. Furthermore we have introduced the ERT encoding scheme for assigning different amounts of redundancy to the different frame types. We have described the advantage of this multilevel redundancy scheme for real time applications and thus it makes sense to prioritize data. At the moment we are running experiments on improving our protocols and to dynamically assign the ERT parameters to the changing of the networking environment, loss rates etc. Also we try to assign more priority levels to MPEG video sequences.

Figure 14: Video sequence from Foreman

Figure 15: Video sequence from Foreman

Acknowledgements

We would like to thank Stephane Boucheron and Andres Desmarck for accounts in Paris and Lund for testing our transfer protocols.

References

[ABEL 94] Albanese, A., Blömer, J., Edmonds, J., Luby, M., *Priority Encoding Transmission*, Technical Report TR-94-039, International Computer Science Institute, Berkeley, 1994.

[BKK+ 95] Blömer, J., Kalfane, M., Karp, R., Karpinski, M., Luby, M., Zuckerman, D., *An XOR-Based Erasure-Resilient Coding Scheme*, Technical Report TR-95-048, International Computer Science Institute, Berkeley, 1995.

[CRM 93] CCITT-Recommendation, MPEG-1, *Coded Representation of Picture, Audio and Multimedia/Hypermedia Information*, ISO/IEC 11172, Geneve Switzerland, 1993.

[CA 94] Chiang, T., Anastassiou, D., *Hierarchical Coding of Digital Television*, Proc. 32^{nd} IEEE Communications Magazine (1994), pp. 38–45.

[Ga 91] Gall, D. L., *MPEG: A Video Compression Standard for Multimedia Application*, Proc. 34, No.4^{th} Comm. ACM (1991), pp. 30–44.

[G 96] Günzel, C., *Fehlerresistente Übertragungssysteme für multimediale Anwendungen*, Diplomarbeit, Institut für Informatik der Universität Bonn, 1996.

[GRW 97] Günzel, C., Riemenschneider, F., Wirtgen, J., *Parallel Real-Time Videotransmission over Lossy Channels*, Proc. 1^{st} Workshop on Cluster - Computing (1997), pp. 231–237.

[Le 94] Leicher, C., *Hierarchical Encoding of MPEG Sequences Using Priority Encoding Transmission (PET)*, Technical Report TR-94-058, International Computer Science Institute, Berkeley, 1994.

[PM 93] Pennebacker, W., Mitchell, J., *JPEG: Still Image Data Compression Standard*, Van Nostrand Reinhold, New York, 1993.

[St 94] Stevens, W. R., *TCP/IP Illustrated, Volume 1*, Addison-Wesley Publishing Company, 1994.

[Wa 91] Wallace, G. K., *The JPEG Still Picture Compression Standard*, Proc. 34, No. 4^{th} Comm. ACM (1991), pp. 46–58.