Parallel Real-Time Videotransmission over Lossy Channels

Christoph Günzel[*] Falko Riemenschneider[†] Jürgen Wirtgen[‡]

Abstract

We present a construction to transmit high quality video streams, such as MPEG-video in realtime over lossy channels. To protect our messages against losses in the network during transit we use a new Forward Error Correction (FEC) scheme called ERT [G 96]. Unfortunately a lot of losses are not caused by the network itself but by buffer overflows in the receivermachine when the receiver is not as fast as the sender. In this case we do not use a single machine as the receiver but a cluster of workstations.

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

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

[‡]Dept. of Computer Science, University of Bonn, 53117 Bonn, Germany. Email: wirtgen@cs.bonn.edu

1 Introduction

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 in our construction. The UDP protocol does not have an error correction mechanism.
Therefore we have to protect our messsage against losses in the network during transmission. Here we use a mechanism called ERT (Error Resilient Transmission). ERT is a Forward Error Correction scheme with several priority levels. It is based on PET (Priority Encoding Transmission) designed at ICSI [BKK^{+} 95] [ABE^{+} 94]. 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. It is useful for applications dealing with realtime transport streams like video or audio in a lossy environment. Such streams for example MPEG-video streams consist of several data parts with different importance [Le 94]. ERT allows to protect those parts with appropriate redundancy and thus guarantees, that the more important parts arrive before the less important ones.
In practice the main problem of using the UDP protocol is the following: When the sender is a very fast server and the receiver is a slow busy client maybe in a LAN-environment we do get a lot of losses caused by buffer overflows in the clientmachine and not by the network. The UDP protocol in its basic does not deal with any synchronization of the sender and receiver machine.
Instead of using one receiver we use a cluster of workstations as the receiver.

2 The TCP/IP protocol

The TCP/IP protocol suite allows computers of all sizes, from many different computer vendors, running totally different operating systems, to communicate with each other. Networking protocols are developed in layers, with each layer responsible for a different facet of communication. A protocol suite such as TCP/IP, is the combination of different protocols at various layers. TCP/IP is normally considered to be a 4-layer system:

1. The link layer, sometimes called the data-link layer or network interface layer, normally includes the device driver in the operating system and the corresponding network interface card in the computer. Together they handle all the hardware details of physically interfacing with the cable.
2. The network layer handles the movement of packets around the network. Routing of packets, for example, takes place here.
3. The transport layer provides a flow of data between two hosts, for the application layer above. In the TCP/IP protocol suite there are two vastly different transport protocols: TCP (Transmission Control Protocol) and UDP (User Datagram Protocol).
4. The application layer handles the details of the particular application.

We are mostly interested in the transport layer. There are two different transport protocols, the TCP and the UDP protocol. TCP provides a reliable flow of data between two hosts. It is concerned with things such as dividing the data passed to it from the application into appropriately sized chunks for the network layer below, acknowledging received packets, setting timeouts to make certain the other end acknowledges packets that are sent, and so on. Although the TCP transport protocol is safe, it is the reason for many delays for realtime applications caused by retransmission of lost data in the channel and therefore it is useless for us.
UDP, on the other hand, provides a much simpler service to the application layer. It just sends packets of data called datagrams from one host to the other, but there is no guarantee that the datagrams reach the other end. Using the UDP transport protocol, we can avoid delays caused by retransmission of lost data. Therefore we have to protect the message against losses ourselves.

3 Erasure Resilient Transmission (ERT)

Using the UDP transport protocol, we have to protect our videostreams against losses in the network during transmission. Here we use a Forward Error Correction scheme called ERT. ERT is an approach to the transmission of prioritized information over lossy packet-switched networks. It is based on PET (Priority Encoding Transmission) designed at ICSI [BKK$^+$ 95] [ABE$^+$ 94]. The basic idea is that the source assigns different priorities to different segments of data. ERT encodes the data using multi-level redundancy and disperses the encoding into the packets to be transmitted. The advantage of ERT is that the receiver is able to recover the data in priority order from any sufficiently large fraction of the transmission. The ERT coding scheme is very useful for the transmission of our videostreams. A MPEG-2 videostream employs three different frame types: intraframe frames (I-frames), predictive frames (P-frames) and bi-directionally interpolated frames (B-frames). I-frames, which are self contained and can be decoded independently, are basically encoded like JPEG pictures [Wa 91]. P-frames need a past I- or P-frame for decoding making use of motion-compensated prediction. In order to code a certain block of a P-frame, the previous I- or P-frame is used as a reference. B-frames are encoded with motion vectors from a past (I- or P-) and a future (I- or P-)frame via interpolative techniques. B-frames are not used for prediction of other frames so the errors in them do not propagate, as opposed to the errors of I- or P-frames. It makes sense to give the highest priority to the I-frames and the lowest priority to the B-frames. The loss of a B-frame in transmission (the receiver does not have a sufficiently large fraction of the transmission to recover it) does not have so many effects on the quality of the video than the loss of an I-frame. The basics of ERT can be explained with the aid of Figure 1. It can be seen that a stream of frames, which constitutes the message to be encoded, is encoded in a code E 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 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. 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 size of the message.

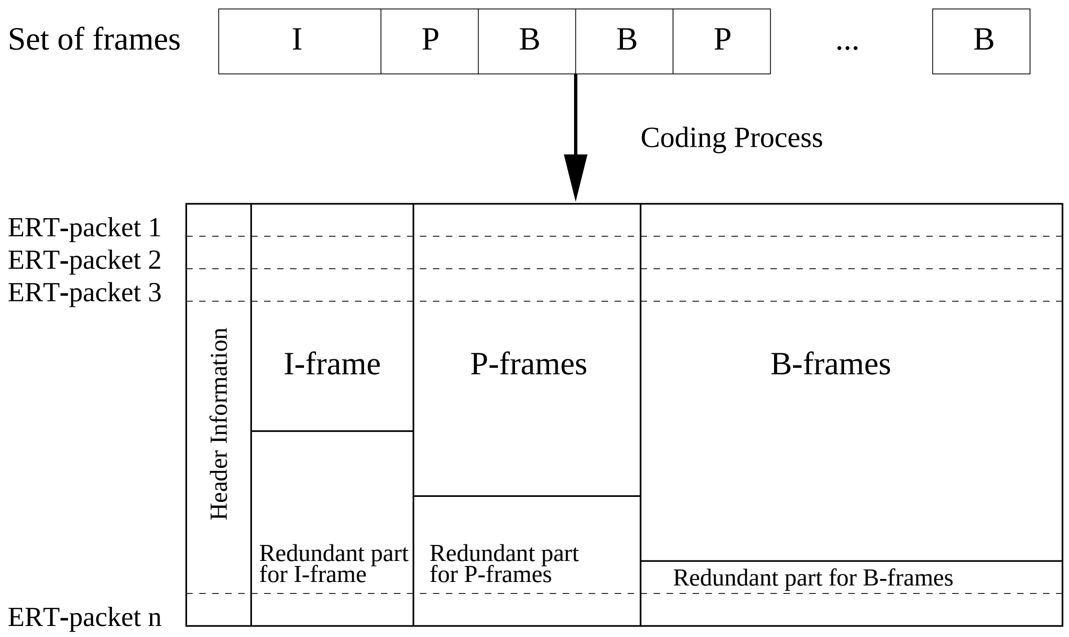


Figure 1: The coding process in ERT

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.

4 Scenery

The ERT encoding scheme is a sufficient solution for packet losses caused by the network in transmit using the UDP transfer protocol. But in practice the main problem is the following: When the sender is a very fast server and the receiver is a slow busy client maybe in a LAN-environment we do get a lot of losses caused by buffer overflows in the clientmachine and not by the network.

To go in more detail we consider the following scenario in our experiments: We assume a company LAN consisting of $n + 1$ workstations, all very busy and slow. Each of the workstations is connected to the internet backbone. The connectivity between the machines is Fast Ethernet Twisted Pair. The network protocol is TCP/IP. Furthermore we assume that the sender of the videosequence is a very fast server. The message should be transferred from the sender to the receiver, a workstation in the company LAN over a lossy media such as the Internet. The whole message is split into jobs for transfer. Each job has for example a size consist of 4 KB of the original message mapped onto 6 Internet packets with a total size of 6 KB using the ERT coding scheme as described above (Figure 1). The transfer is done via the UDP protocol to avoid delays caused by retransmissions of lost data.

The processes which are running on the senderworkstation are the following:

1. Read the videosequence from a file or a camera
2. Encode the videosequence with the ERT-encoder for protection against losses
3. Send the encoded sequence to the receiver

On the receiverworkstation the processes are similar:

1. Receive the encoded sequence
2. Decode the videosequence with the ERT-decoder
3. Play the videosequence

As mentioned above we get a lot of losses of data caused by buffer overflows in the receivermachine and not by the network. To avoid these losses we do not use a single workstation in the LAN as the receiver but we do use a cluster of workstations for the receiving and decoding process.

5 Parallel Real-Time Transmission

To avoid losses caused by buffer overflows in the receivermachine we do not use a single workstation in the LAN as the receiver but we do use a cluster of workstations for the receiving and decoding as described in Figure 2.

The sender transmits the message not to a single receiver but to n receivers. The processes running on the sendermachine are the same. Only the sending of the encoded sequence

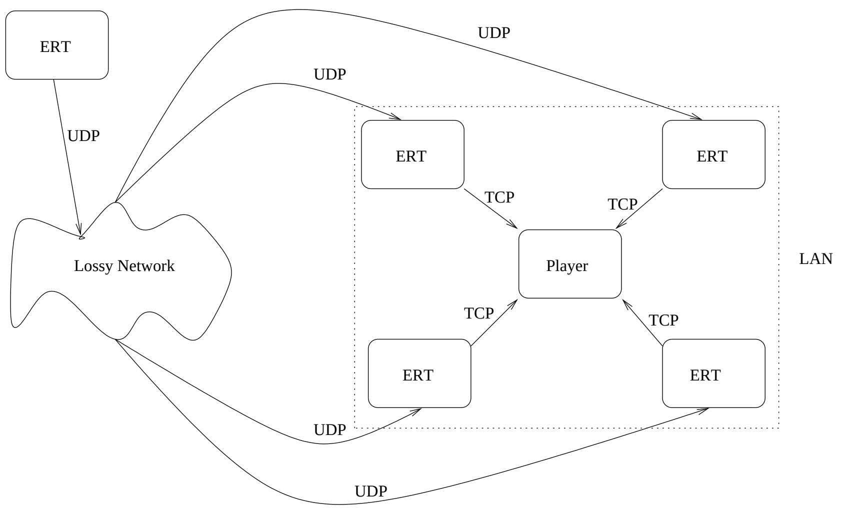


Figure 2: Scenario with $n = 4$ receivers

is a bit different. Now the sender has n receivers with different port addresses. We send the job with the number i to the receiver (i modulo n). On the receivermachines the processes differ. Consider receiver k:

1. Receive the encoded message (jobs with jobnumbers are multiples of k)
2. Decode the message using the ERT-decoder
3. Send the decoded message via the TCP/IP protocol to the 'Player'

Finally the processes running on the player are the following:

1. Receive the decoded jobs from the n receivers
2. Play the jobs in order to their jobnumbers

The results of the parallizing of the receiving and decoding processes are shown in the next chapter.

6 Experimental Results

This chapter contains the results of some of our experiments. The sendermachine was installed in Lund, Sweden, the receivermachine in Bonn. The sender was a very fast SUN Ultra 1 workstation, the receivers in the cluster were old slow SUN SPARC 1 stations. The connection between Bonn and Lund was excellent, so only a few losses were caused by the Internet itself. The transfertime of the message, the MPEG-videostreams, was around 20 minutes. The message was split into jobs containing 8 packets, as explained in chapter 3. Packets $0 - 3$ of each job contain the cleartext information of the I-frames, packets $0 - 5$ the P-frames and packets $0 - 6$ the B-frames. Figure 3 shows the number of received packets per job depending on the number of receivers.

workstations in cluster	1	2	3	4
receiver / player	1	1 / 1	2 / 1	3 / 1
number of jobs send	16562	17684	17688	17688
jobs with 0 packets received	4214	781	794	48
jobs with 1 packet received	2158	738	166	8
jobs with 2 packets received	2140	1450	212	9
jobs with 3 packets received	2188	2795	318	7
jobs with 4 packets received	2036	3904	383	4
jobs with 5 packets received	1608	3943	400	7
jobs with 6 packets received	1183	2294	513	11
jobs with 7 packets received	638	775	843	129
jobs with 8 packets received	397	1004	14069	17469

Figure 3: received packets per job

In Figure 4 to 7, we can see the changing of the quality of the MPEG-video with increasing numbers of receivers. The quality of the video depends on the number of packets received per job. If the number of packets per job is at most three nothing can be decoded and the quality is bad. For jobs containing 4 to 5 packets, the quality is poor because only the I-frames can be decoded. If the number of packets per job is 6 the quality is fair. I- and P-frames can be decoded. For jobs containing at least 7 packets, the quality is good, because all frames can be decoded. In the case you have only one receiver (Figure 4) which is simultaniously the player of the movie the quality of the movie is bad in 64.6% of the received jobs, poor in 22%, fair in 7.14% and good in only 6.25% of the received jobs. When you have a second machine, so that the receiving and playing processes are separated (Figure 5), the quality of the movie is bad in 32.42% of the received jobs, poor in 44.38%, fair in 13% and good in 10.06% of the received jobs. When you have two machines as the receivers and one as the player (Figure 6), the increasing of the quality of the movie is overwhelming. The quality is bad in only 8.43% of the received jobs, poor in 4.43%, fair in 2.9% but good in 84.22% of the received jobs. We can improve this result by using more than two receiver (Figure 7). The quality of the movie is going very close to its optimum in this case.

7 Conclusion and further research

With the parallel receiving and decoding of the jobs in our cluster we get a sufficient and practicable solution for our problem (Figure 6, 7). For many application such as video on demand, audio on demand or videoconferencing, our construction is a useful method if the receivermachine is not as powerful as it should be to deal with realtime applications. The use of a cluster of $n + 1$ workstations as the receiver for a transmission is just an example. Furthermore one can use such a cluster as sender or on both sides the sending and receiving side. At the moment we run experiments on a dynamization of the sizes of such clusters. We are looking for a solution for a dynamic variation of the number of machines envolved in a cluster for sending on the loadbalance of each machine.

Acknowledgements

We thank Stephane Boucheron, Elias Dahlhaus and Anders Dessmark for accounts in Paris (France), Köln (Germany) and Lund (Sweden) for our experiments. We also thank Carsten Dorgerloh for helpful comments and interesting discussions.

References

[ABE+ 94] Albanese, A., Blömer, J., Edmonds, J., Luby, M., Sudan, M., *Priority Encoding Transmission*, Proc. 35th IEEE FOCS (1994), pp. 604–613.

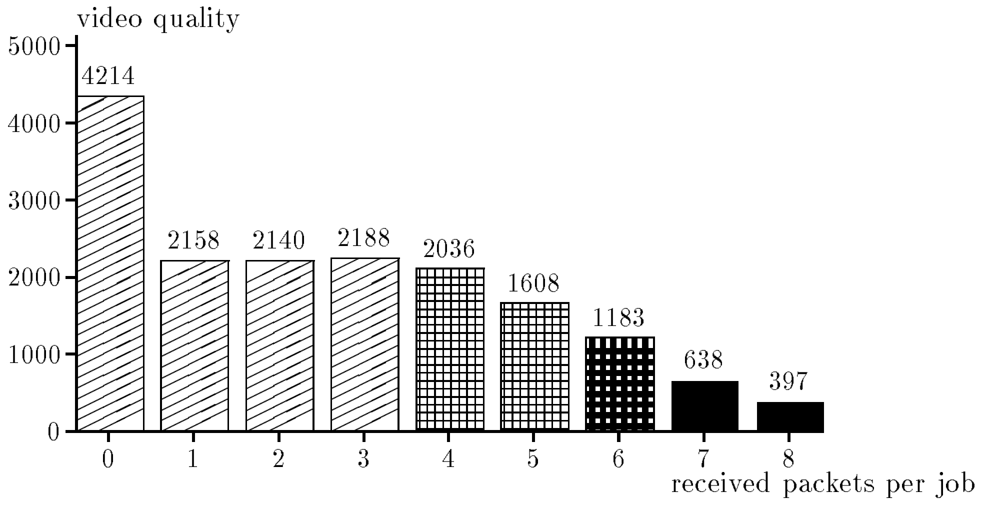
[BKK+ 95] Blömer, J., Kalfane, M., Karp, R., Karpinski, M., Luby, M., Zuckerman, D., *An XOR-Based Erasure-Resilient Coding Scheme*, Technical Report TR-95-048, International Computer Science Institute, Berkeley, 1995.

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

[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.

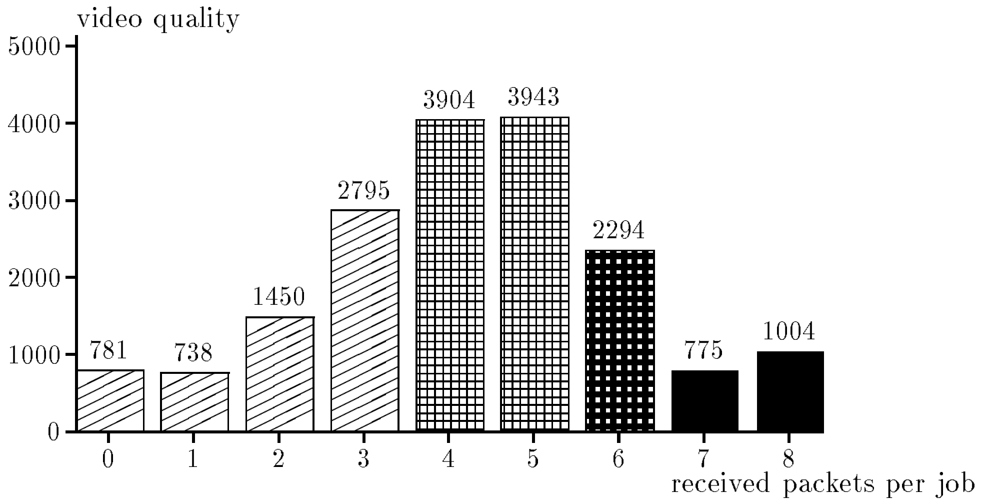
[St 94] Stevens, W. R., *TCP/IP Illustrated, Volume 1*, Addison-Wesley Publishing Company, 1994.

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



Legend: ▨ bad ▦ poor ▩ fair ■ good

Figure 4: one machine in cluster



Legend: ▨ bad ▦ poor ▩ fair ■ good

Figure 5: two machines in cluster

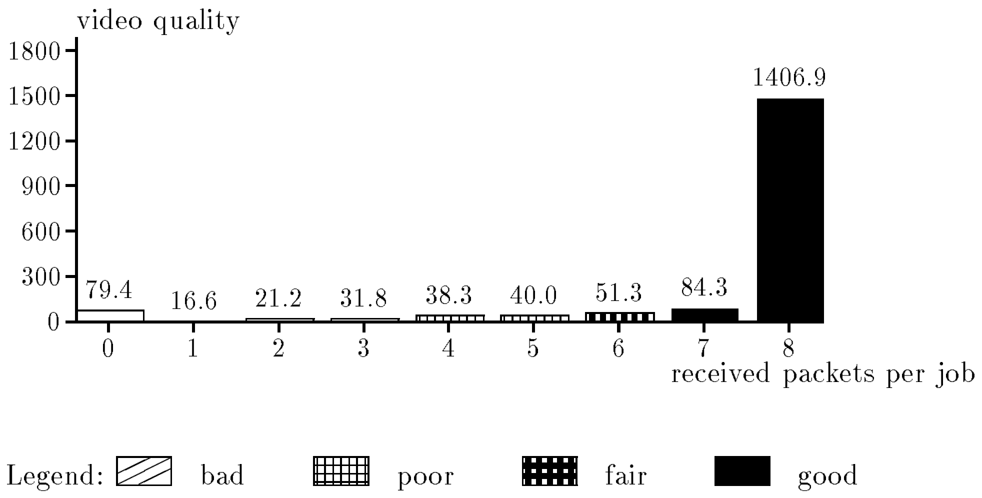


Figure 6: three machines in cluster

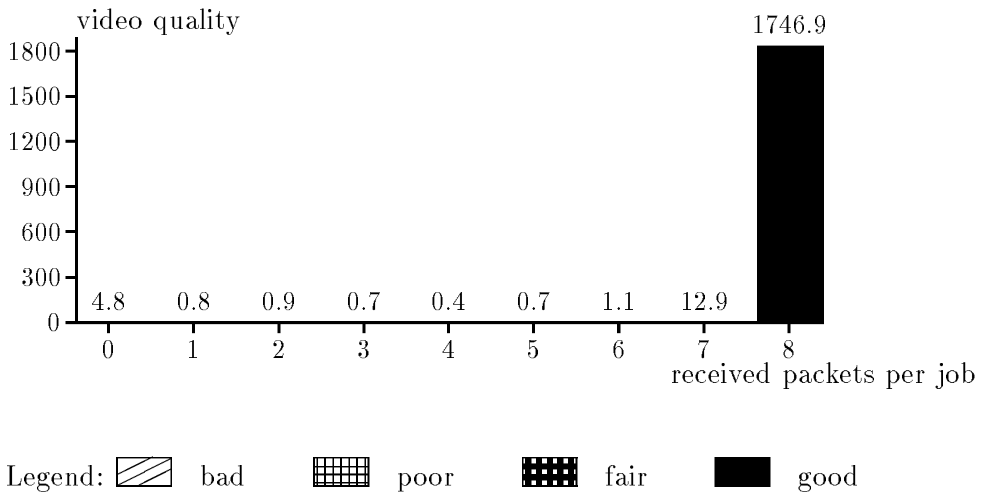


Figure 7: four machines in cluster