

Improved Videotransmission over Lossy Channels using Parallelization

Christoph Günzel¹, Falko Riemenschneider¹, and Jürgen Wirtgen¹

Dept. of Computer Science, University of Bonn, Germany. Email:
{guenzel,riemenc,wirtgen}@cs.bonn.edu

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 Erasure Resilient Transmission (ERT). Unfortunately a lot of losses are not caused by the network itself but by buffer overflows in the receiver machine when the receiver is slower as the sender. Therefore the receiving is not done on a single machine but parallel on a cluster of workstations.

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 [6] 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 [6] protocol in our construction. 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). ERT is a Forward Error Correction scheme with several priority levels. It is based on [2] [1] [4]. 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 [5]. ERT allows to protect those parts with appropriate redundancy and thus guarantees, that the more important parts arrive before the less important ones.

The other 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 also get losses caused by buffer overflows in the client 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 Erasure Resilient Transmission (ERT)

Using the UDP protocol, we have to protect our videostreams against losses in the network during transmission. Therefore we use a Forward Error Correction scheme called Erasure Resilient Transmission (ERT). ERT is an approach to the transmission of prioritized information over lossy packet-switched networks. It is based on [2] [1] [4]. 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. In more detail: Given a message $M = (M_1, M_2, \dots, M_m)$, -the message M is split in m packets for transmission-, the ERT encoder generates a Code $C = (C_1, C_2, \dots, C_n)$ 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. The ERT coding scheme is very useful for the transmission of our videostreams. A MPEG videostream assumes three different frame types: intraframe frames (I-frames), predictive frames (P-frames) and bi-directionally interpolated frames (B-frames). These frames are organized in so-called group of pictures (GOPs). Every GOP starts with an I-frame, which is encoded independent of other frames and thus can be displayed independently of the content of other frames which is important to limit effects of error propagation within the total video stream [7]. The other frames in a GOP are P-frames, which are predicted taking into account the content of the last I- or P-frame and thus are dependent on the correctness of those frames. The B-frames are bi-directionally interpolated based on the content of the two neighbouring I- resp. P-frames. 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 does not have so many effects on the quality of the video than the loss of an I-frame. (see [3] for further details on the MPEG standard) In Figure 1 the generation of the code is explained 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 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.

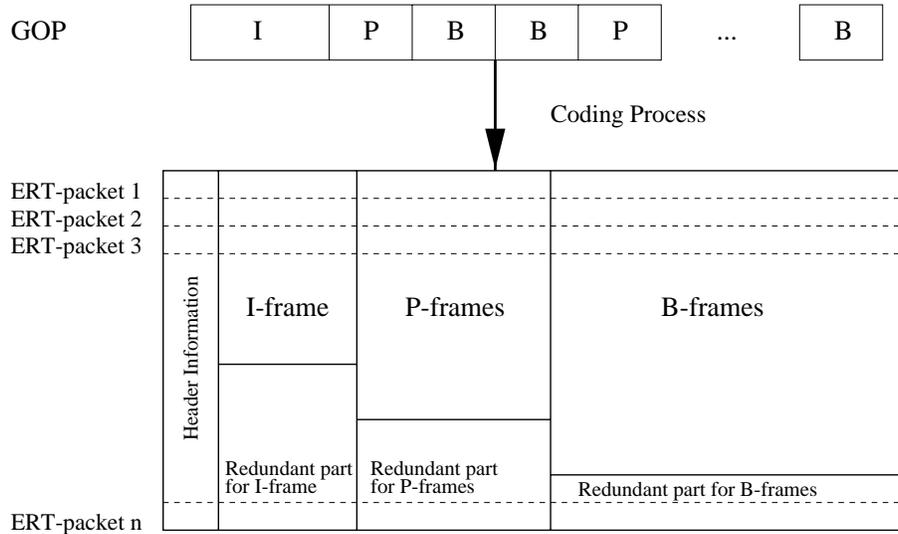


Fig. 1. The coding process in ERT

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.

3 Real-Time Transmission

The ERT encoding scheme is a sufficient solution for packet losses caused by the network in transit using the UDP transfer protocol. Another problem that occurs 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 client machine and not only by the network. We have to protect the message against losses caused by the network and caused

by bufferoverflows. A solution could be the following: We increase the amount of redundancy generated by the ERT encoder. Therefore we had to send more packets. This would increase the overhead given to the original message. We would need more bandwidth to transfer the packets in the same time as unprotected. The receiver has to collect more packets and we get more bufferoverflows and so on. Our goal is to avoid the increasing of traffic caused by generating more redundant information than needed to protect our message against losses during transit.

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 video consists of GOPs of different sizes. Each GOP is protected against losses in the network using the ERT coding scheme as described above (Figure 1). The encoded GOP is split into jobs of 8 Internet packets of size 1 KB for transfer. 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 GOP per GOP
2. Encode the GOP with the ERT-encoder for protection against losses
3. Send the encoded GOP to the receiver

On the receiverworkstation the processes are similar:

1. Receive the encoded GOP
2. If possible, decode the GOP with the ERT decoder
3. Play the GOP

As mentioned above we get a lot of losses of data caused by buffer overflows in the receiver machine 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.

4 Parallel Real-Time Transmission

To avoid losses caused by buffer overflows in the receiver machine 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 GOPs not to a single receiver but to n receivers. The processes running on the sender machine are the same. Only the sending of the encoded GOPs is a bit different. Now the sender has n receivers with different port addresses. We send the jobs containing the GOP with the number i to the receiver (i modulo n). On the receiver machines the processes differ. Consider receiver k :

1. Receive the encoded GOP (jobs containing GOPs with numbers are multiples of k)

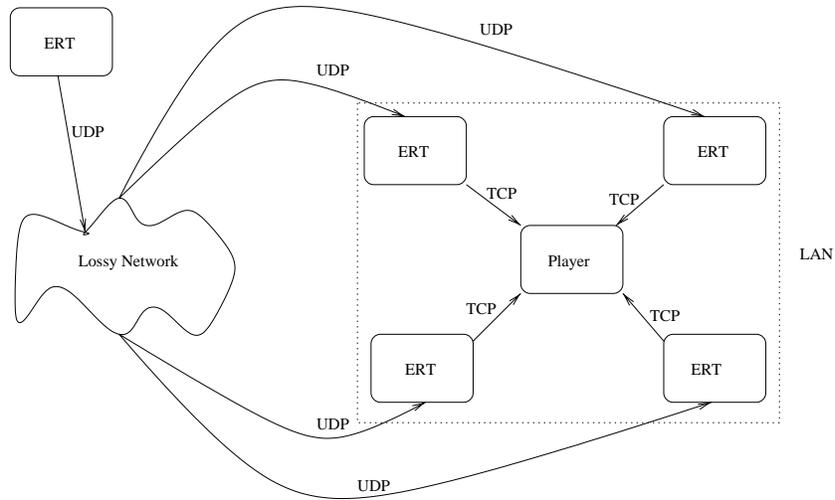


Fig. 2. Scenario with $n = 4$ receivers

2. Decode the GOP using the ERT-decoder
3. Send the decoded GOP via the TCP/IP protocol to the 'Player'

Finally the processes running on the player are the following:

1. Receive the decoded GOPs from the n receivers
2. Play the GOPs in order to their numbers

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

5 Experimental Results

This chapter contains the results of some of our experiments. The sender machine was installed in Lund, Sweden, the receiver machine in Bonn. The sender was a fast SUN Ultra 1 workstation, the receivers in the cluster were old slow SUN SPARC 1 stations. The length of our video was around 20 minutes. We sent the video "Red's Nightmare" in a loop 40 times. "Red's Nightmare" contains 41 GOPs including 41 I-frames, 81 P-frames and 1085 B-frames. The size of the total video is 3,62 MB. The average size of a GOP is 88 KB. Before the transfer we observed the network conditions with a test protocol. The result was the following: We had losses up to 40% between Lund and Bonn. To protect the video against these losses we added 70% redundancy to I-frames, 50% redundancy to P-frames and 25% redundancy to B-frames. This leads to a total overhead of around 35% of the original GOPs. The receiver needed fractions of 60% to recover I-frames, 75% to recover P-frames and 87% to recover B-frames. For realtime we need 150 KB/sec bandwidth for this video. For the encoded video we now

need 210 KB/sec. But this was no problem between Bonn and Lund. The GOPs were split into jobs containing 8 packets of size 1 KB, as explained in chapter 3. Figure 3 shows the number of received packets per job depending on the number of receivers.

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

Fig. 3. received packets per job

In Figure 4 to 6, we can see the distribution of the losses per job. The quality of the video depends on the number of packets received per GOP. As mentioned above, the average size of a GOP is 88 KB. We add redundancy and get the encoded GOP of size 120 KB. The GOPs are split into 15 jobs containing 8 packets of size 1 KB. For decoding the I-frames, the receiver needs 60% of the packets (72), for the P-frames 75% (90), for B-frames 87% (105). In scenario Figure 4 the sender sends 16562 jobs to the receiver. The receiver gets 43926 packets. That means the receiver gets 2,7 packets per job in average. For each GOP he receives 41 packets. The quality of the video is very bad. In scenario Figure 5 the sender sends 17684 jobs containing 8 packets to the receiver. The receiver is able to get 74575 packets. Thus the receiver gets 4,5 packets per job in average. For each GOP he receives 63,3 packets. This is sufficient to recover the I-frames. The quality of the video is fair. In scenario Figure 6 the sender sends 17688 jobs to the receiver. The receiver gets 126607 packets. That means the receiver gets 7,1 packets per job in average. For each GOP he receives 107 packets. The receiver is able to recover all frame types now. The quality of the video is excellent.

6 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). For many application such as video on demand, audio on demand or videoconferencing, our construction is a useful method if the receiver machine 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 involved 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

1. A. Albanese, J. Blömer, J. Edmonds, M. Luby, and M. Sudan. Priority encoding transmission. In *IEEE FOCS*, volume 35, pages 604–613, 1994.
2. J. Blömer, M. Kalfane, R. Karp, M. Karpinski, M. Luby, and D. Zuckerman. An xor-based erasure-resilient coding scheme. Technical Report TR-95-048, International Computer Science Institute, Berkeley, August 1995.
3. D. Le Gall. Mpeg: A video compression standard for multimedia application. In *Comm. ACM*, volume 34, No.4, pages 30–44, 1991.
4. C. Günzel. Fehlerresistente übertragungssysteme für multimediale anwendungen. Diplomarbeit, Institut für Informatik der Universität Bonn, 1996.
5. C. Leicher. Hierarchical encoding of mpeg sequences using priority encoding transmission (pet). Technical Report TR-94-058, International Computer Science Institute, Berkeley, 1994.
6. W. R. Stevens. *TCP/IP Illustrated, Volume 1*. Addison-Wesley Publishing Company, 1994.
7. G. K. Wallace. The jpeg still picture compression standard. In *Comm. ACM*, volume 34, No. 4, pages 46–58, April 1991.

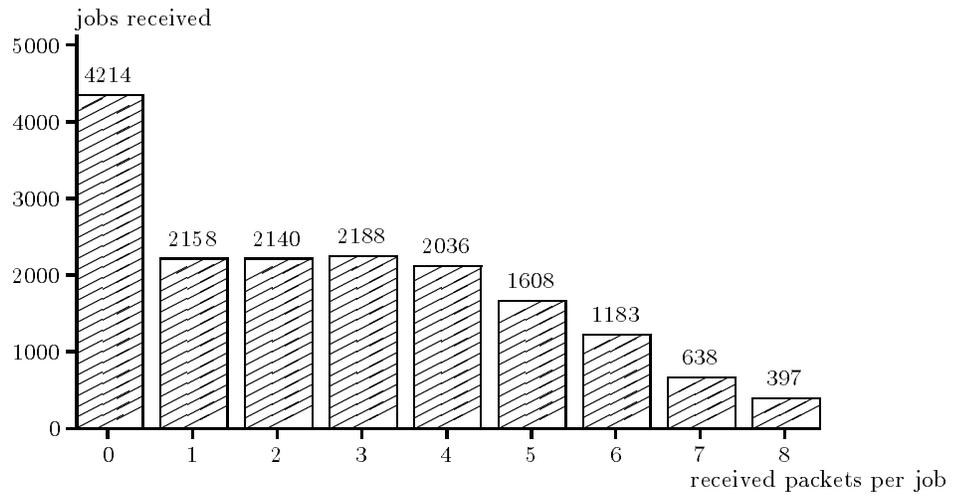


Fig. 4. one machine in cluster

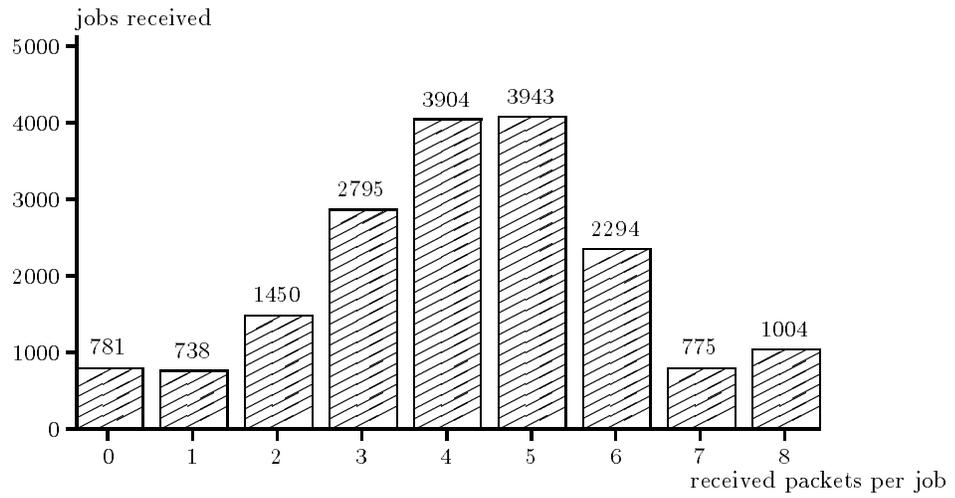


Fig. 5. two machines in cluster

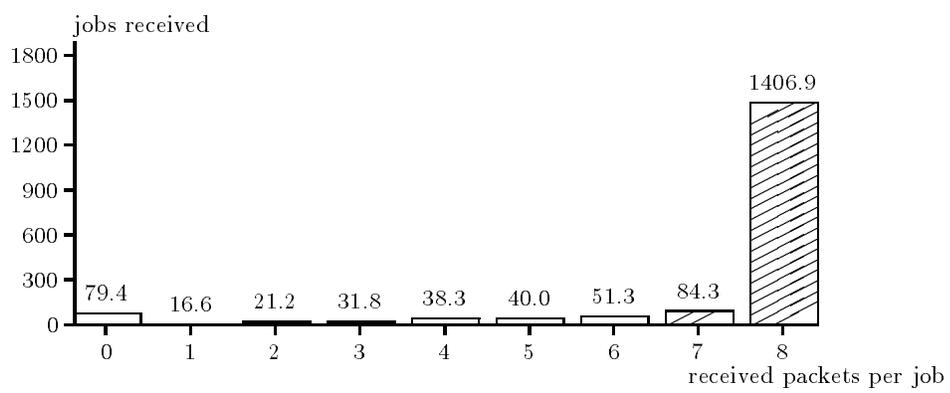


Fig. 6. three machines in cluster