NTNU
Innovation and Creativity

Feedback-based Error Control Methods for H.264

Stian Selnes

Master of Science in Communication Technology
Submission date: June 2007
Supervisor: Andrew Perkis, IET
Co-supervisor: Yuan Lin, Q2S

Norwegian University of Science and Technology
Department of Electronics and Telecommunications

Problem Description

Video traffic is expected to be an important component of packet network traffic. Quality of service (QoS) guarantees are required in order to obtain persistent perceptual quality. However, video codecs with fixed parameter settings do not adapt to time-varying network characteristics. Video applications that are aware of channel characteristics will outperform those who are not, and network feedback information through backward channels play an important role in improving network QoS. The thesis is to develop an error resilience scheme for H.264 utilizing feedback from the decoder. This implies an examination of both the feedback mechanism and how to employ the received information in the video coding process, in order to improve the error robustness. The former shall be the main focus of this thesis. It is desirable to apply techniques that conform to existing standards and are applicable for video communication systems. The complete scheme, including the feedback mechanism and encoder operations, shall be implemented in the H.264 reference software. The developed methods shall be evaluated by comparing them to existing resilience schemes. The comparisons shall be carried out by real-time simulations in an IP network test bed.

Assignment given: 24. January 2007
Supervisor: Andrew Perkis, IET

Summary

Many network-based multimedia applications transmit real-time media over unreliable networks, i.e. data may be lost or corrupted on its route from sender to receiver. Such errors may cause a severe degradation in perceptual quality. It is important to apply techniques that improve the robustness against errors, in order to ensure that the receiver is able to playback the media with the best attainable quality.

Today, most Error Robustness (ER) schemes for video employ proactive error resilient encoding. These schemes add redundant information into the encoded video stream in order to increase the robustness against potential errors. Because of this, most proactive schemes suffer from a significant reduction of the coding efficiency. Another approach is to adjust the encoder operations based on feedback information from the decoder, e.g. to repair corrupted regions based on reports of lost data. Feedback-based ER schemes normally improves the coding efficiency compared with proactive schemes. Moreover, they adjust rapidly to time-varying network conditions.

The objective of this thesis is to develop and evaluate a feedback-based ER scheme conforming to the H.264/AVC standard and applicable for real-time low-delay video applications. The scheme is referred to as Feedback-Based Intra Refresh (FBIR). The performance of FBIR will be compared with an existing proactive ER scheme, known as Intelligent Packet Loss Recovery (IPLR). Special attention is given to the applied feedback mechanism, Extended RTP Profile for RTCP-based Feedback (RTP/AVPF).

RTP/AVPF is a new (2006) feedback protocol. Basically, it specifies two modifications/additions to the Real-time Transport Control Protocol (RTCP): First, it modifies the timing algorithm to enable early feedback, while not exceeding the RTCP bandwidth constraint. Second, new RTCP message types are defined, which provides information useful for error control purposes.

FBIR employs RTP/AVPF to provide timely feedback of lost packets from the decoder to the encoder. Upon reception of this feedback, the encoder use a fast error tracking algorithm to locate the erroneous regions. Finally, the regions that are assumed to be visually corrupted after decoding are intra refreshed.

IPLR is an ER scheme developed for use in a commercial video communication system. It applies a motion-based intra refresh routine.

The comparison is carried out by online simulations with various network environments (0, 1, 3 and 5% loss rate; 50 and 200 ms latency), bit rates (64, 144 and 384 kbit/s) and video sequences. First, the video is encoded and transmitted in real-time to the decoder via a network emulator. This emulator generates the desired network characteristics. The receiver decodes the video in real-time and transmits feedback information back to the encoder. The encoder adjusts its encoding process according to this feedback. The H.264/AVC reference software is modified and used as codec. Finally, objective quality measures are obtained by calculating the Peak Signal-to-Noise Ratio (PSNR) of the decoded videos. In addition, some visual inspection is performed.

Isolated measures on the RTP/AVPF transmission algorithm are also performed. These show that RTP/AVPF is able to provide timely feedback for error control purposes for a great number of applications and network environments. However, the experienced feedback delay may be increased by numerous factors, e.g. the network latency, the packet loss rate, the session bandwidth, and the number of receivers. This may decrease the performance of ER schemes utilizing RTP/AVPF.

RTP/AVPF is fairly easy to implement since it only modifies the RTCP timing algorithm and adds new RTCP message types. RTP/AVPF may be used in combination with other standards in order to extend the available feedback information. Hence, RTP/AVPF enables timely feedback for use in a wide range of multimedia applications.

The PSNR measurements show that FBIR always obtains higher objective quality than IPLR for error free transmissions. This does not, however, necessarily affect the perceptual quality if the bit rate is high. FBIR achieves higher PSNR in other situations as well, such as for very low loss rates, low or medium bit rates, and for sequences with high or medium motion activity. Conversely, IPLR performs better for low motion sequences encoded at high bit rates when the loss rate exceeds a certain threshold, typically about 1%. It is also shown that the performance of FBIR may be reduced if the network latency increases. Visually, the main difference between the two schemes is that FBIR recovers all corrupted regions at one instant, while IPLR performs a gradual refresh. The average time before recovery is somewhat shorter for IPLR.

The differences between FBIR and IPLR are mainly caused by two factors. First, using FBIR results in less intra coding and thus better coding efficiency. Second, the FBIR scheme does not repair errors until the encoder receives the feedback. Usually, this happens *after* IPLR has repaired most of the corrupted region. In short, one can say that FBIR provides medium error robustness and high coding efficiency, in contrast to IPLR's high robustness and low coding efficiency. While FBIR's performance may be reduced by network characteristics such as increased latency, IPLR is unaffected by these factors.

For error free transmissions, FBIR does not significantly reduce the coding gain compared with a non-robust encoding scheme. Still, it provides a good robustness against corruption in error-prone networks. Thus, all real-time video systems that benefit from immediate feedback should strongly consider to employ FBIR or similar feedback-based ER schemes.

Preface

This master's thesis was submitted to the Norwegian University of Science and Technology (NTNU), Department of Electronics and Telecommunications (IET). It represents the end of a five year study for the Master of Science Degree in Communication Technology, Multimedia Signal Processing.

First, I would like to express my gratitude to Yuan Lin for being a dedicated supervisor, for her ideas and guidance throughout the work of the thesis. Second, I would like to thank Odd Inge Hillestad for his help on setting up the testing environment and for feedback during the writing process. I would also like to thank Arild Fuldseth at TANDBERG for providing details on their error resilience scheme, IPLR. Finally, I would like to thank professor Andrew Perkis for giving advices and supervising the process, and for defining some aspects of the problem description according to my own professional interests.

Trondheim, June 21, 2007

Stian Selnes

Contents

List of Figures

List of Tables

Acronyms

ACK	ACKnowledgement
ARQ	Automatic Repeat reQuest
AVPF/CCM	Codec Control in the RTP Audio-Visual Profile with Feedback
CDF	Cumulative Distribution Function
CR	Contamination Ratio
ER	Error Robustness
FB	FeedBack
FBIR	Feedback-Based Intra Refresh
FEC	Forward Error Correction
fps	frames per second
GOB	Group Of Blocks
HVS	Human Visual System
IETF	Internet Engineering Task Force
IDR	Instantaneous Decoding Refresh
IP	Internet Protocol
IPLR	Intelligent Packet Loss Recovery
JVT	Joint Video Team
LAN	Local Area Network

MB	MacroBlock
MTU	Maximum Transmission Unit
MV	Motion Vector
NACK	Negative ACKnowledgement
NAL	Network Abstraction Layer
PSNR	Peak Signal-to-Noise Ratio
QoS	Quality of Service
RPS	Reference Picture Selection
RR	Receiver Report
RTCP	Real-time Transport Control Protocol
RTP	Real-time Transport Protocol
RTP/AVPF	Extended RTP Profile for RTCP-based Feedback
SDES	Source DEScription
SR	Sender Report
TCP	Transmission Control Protocol
UDP	User Datagram Protocol
VBCM	Video Back-Channel Message
VCL	Video Coding Layer
Y-PSNR	Luminance PSNR

CHAPTER 1

Introduction

There has been a tremendous development in the field of video coding and related technologies during the last one and a half decade. Since 1990 when H.261 was standardized, standards such as MPEG-1, H.262/MPEG-2, H.263, MPEG-4 and H.264/MPEG-4 Part 10 has improved the video coding scheme. The gain has primarily been with respect to coding efficiency, but also the resistance to errors has been improved. Not only has this benefited existing services and applications, but also spawned brand new industries and services. Video conferencing, video-on-demand, video streaming, high-definition TV, video recording and video on mobile devices are examples of technologies that have gained from this development. The importance of such services will grow even more in the future. The performance and use of multimedia and entertainment devices, Internet and computers in general will increase and more people will get access to this technology.

Generally, users have a set of expectations and requirements to all services they use. If a service fail to fulfill these requirements, the user will consider it as being of poor quality. This applies also to multimedia services. The requirements for many of these, such as audio and video applications, are high since they may be compared to “real-life quality”. For some systems, especially for network-based systems utilizing real-time encoding and decoding of the media, it may be difficult to meet these requirements. Therefore, it is often necessary to take adequate measures in order to improve the end-user perceived quality.

A real-time video system involves several steps. First, the video is compressed by an encoder to reduce the data rate while keeping an acceptable visual quality. The compressed bit stream is then multiplexed, packetized and channel encoded prior to transmission over a network. The receiver performs similar reverse operations and feeds the resulting bit stream into the video decoder to reconstruct the video. If the network does not guarantee error free transmission, data may be lost and cause the receiver to experience a degradation in quality. It is desirable to either eliminate the probability of data loss, or to minimize the effect of such losses to preserve a decent quality. The system's ability to recover from lost or corrupted data is known as error robustness.

Various robustness techniques exist for the different coding standards. However, they all fit in one of three main categories: proactive error resilient encoding, error concealment or feedback-based error control. Because real-time multimedia applications suffer from very strict timing requirements, most existing error robust systems utilize a combination of proactive error resilient coding and error concealment. The latter is performed within the decoder upon detection of errors in order to conceal and reduce their impact. Proactive techniques add redundant information into the video stream. This improves the robustness against *potential* errors but also decreases the compression efficiency. Herein lies the main drawback of proactive Error Robustness (ER) schemes: they may waste bits to protect information that is not a subject to error. The third group of robustness techniques, feedback-based error control, address this problem. Such techniques set up a feedback channel for the decoder to report information about the received data and current network conditions. This helps the encoder to rapidly adjust its operations according to the decoder's status. For instance, the feedback may be used to recover reported errors. Thus, the encoder may choose to only use additional bits to recover from *actual* errors. This generally improves the coding efficiency compared to proactive ER techniques, but with respect to robustness the performance depends heavily on the tools applied.

The objective of this thesis is to develop and evaluate a feedback-based ER scheme, referred to as Feedback-Based Intra Refresh (FBIR), conforming to the H.264/AVC video coding standard [1] and applicable for real-time low-delay video applications, such as conversational video systems. Its performance will be objectively compared with an existing proactive scheme, known as Intelligent Packet Loss Recovery (IPLR) [2] [3], and be subject to visual inspection. The schemes are evaluated by performing online simulations in an IP testbed, configured to simulate a various set of lossy network environments. The feedback mechanism in particular will be subject for an in-depth examination.

Extended RTP Profile for RTCP-based Feedback (RTP/AVPF) [4], a modification of Real-time Transport Control Protocol (RTCP) [5] allowing immediate feedback within the RTCP bandwidth constraint, will be employed to report lost data. The error propagation of reported losses will be tracked using a fast tracking algorithm [6]. At last, intra refresh is performed on regions that are assumed to not belong to a static area, since static areas normally are very well concealed by the decoder.

The thesis is organized as follows. First, in Section 2, background theory for the following experiments and discussion is presented. This includes theoretical information about some properties of H.264/AVC and its tools of value for this study, a presentation of error robustness in general, possible feedback methods, and related research on feedback-based error control. The section continues with a detailed description of the developed ER scheme, FBIR. Section 3 explains the methodology of the study. In Section 4 the results from the simulations and comparisons of the ER schemes are presented. Discussion of the obtained results and an evaluation of FBIR and RTP/AVPF are given in Section 5. Finally, some plans for future work are suggested before the report is concluded in Section 6 and 7, respectively.

CHAPTER 2

Background and theory

This section first provides a high-level overview of the digital video coding standard H.264/AVC, its encoding scheme and usage in packet-switched networks. An introduction to general error robustness techniques is given. In addition, the ER tools available for H.264/AVC that are relevant for this study are presented. After this, different available feedback mechanisms are described. Then, related works within the field of error resilience and feedback-based error control are presented, before FBIR and IPLR are explained in detail. Finally, a brief description of the H.264/AVC reference software and the applied quality assessment, Peak Signal-to-Noise Ratio (PSNR), is given.

2.1 H.264/AVC

ITU-T H.264 / MPEG-4 (Part 10) Advanced Video Coding [1] (commonly referred to as H.264/AVC or only H.264; the latter will be used through the rest of this thesis) was approved as a video coding standard by Joint Video Team (JVT)[1] in mid 2003. Three of several imposed requirements for this standard were to increase the compression performance, improve the network friendliness and enable enhanced error and packet loss resilience tools compared to previous standards [7]. H.264 is applicable in a wide range of applications, from real-time communication to high quality consumer and broadcast systems. The following description of the standard will focus on features relevant for real-time low-delay applications, such as conversational video systems. It is important to be aware of that only the central decoder is standardized by imposing restrictions on the bit-stream and syntax. Thus, the encoder has great flexibility in how to apply the various coding tools to achieve optimality for the current application.

[1]JVT is a collective partnership between ITU-T Video Coding Experts Group and the ISO/IEC Moving Picture Experts Group (MPEG).

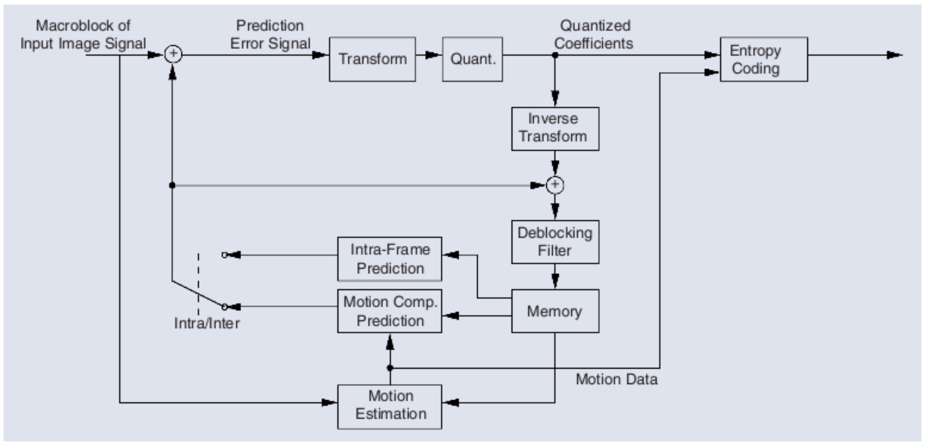


Figure 2.1: High-level H.264 encoder architecture [9].

2.1.1 The H.264 encoder

To efficiently compress a video the H.264 standard uses a design principle similar to prior standards since H.261. These coders are commonly referred to as hybrid coders because they exploit both temporal and spectral redundancy in the encoding process. Moreover, H.264 utilizes spatial redundancy. Figure 2.1 illustrates the high-level architecture of an H.264 encoder. The encoding process is block-based. That is, each video frame is divided into non-overlapping MacroBlocks (MBs) of 16x16 luma samples and their corresponding chroma components, which are used as the basic units in the compression scheme. The following information is collected from [8] [9].

To reduce spatial redundancy intra-frame prediction is used. The encoder predicts the current MB or its 4x4 sub-blocks from spatially neighboring samples. A fixed set of different prediction modes are supported, and the one which gives the least prediction error is chosen.

Temporal redundancy is reduced through inter-frame prediction. The encoder tries to estimate the current MB based on information in previous and/or future frames. More precisely, motion estimation from allowed reference frames is performed on the MB as a whole or on smaller blocks derived from the MB. This estimation is calculated with quarter-sample accuracy. The best match is then represented by the prediction error signal and motion data (Motion Vectors (MVs) and reference frame parameters).

The decision on which prediction technique should be applied is normally performed at MB level. However, there are some restrictions on this mode decision process. Groups of MBs are logically segmented into slices, which may contain as little as one MB or as much as a whole frame. The slice type, I (intra), P (predicted) or B (bi-predicted), is decisive for if and how the inter prediction may be performed. In I-slices none of the MBs are inter predicted. P-slices allow inter prediction with at most *one* motion-compensated prediction signal per prediction block besides the modes available for I-slices. B-slices allow *two* prediction signals per prediction block, as well as the coding types available for P-slices. Each slice is independently decodable from the other slices in the same frame, assuming that the active sequence and picture parameter set along with the reference pictures for the slice are available. This implies that intra prediction is not allowed across slice boundaries. A frame that only consists of I-slices is called an I-frame.

For both intra and inter prediction, the prediction error signal is transformed to the spectral domain. This enables more efficient quantization and entropy coding. For every coded frame, the encoder transmits the transformed coefficients, motion data and header information essential for decoding.

In addition to the prediction schemes above, special modes are available for P- and B-slices, *skip mode* and *direct mode*, respectively. These are to be used for picture areas with no change or constant motion, typically a static background or slow panning. For such MBs, neither a quantized prediction error signal nor a motion vector is transmitted. Instead, these MBs are decoded using only already received information.

Despite the similar high-level design, there are many differences relative to prior standards that add up and improve the coding efficiency for H.264 significantly. Some of the most important ones are enhanced motion-prediction capability, use of a small block-size exact-match transform, adaptive in-loop deblocking filter and enhanced entropy coding methods [8]. More precisely, the motion compensation may use variable and small block-sizes, quarter-sample accuracy, multiple reference pictures and weighted prediction; the transformation take advantage of 4x4 transforms instead of 8x8 as in prior standards and integer transforms are used to avoid inverse-transform mismatch; an in-loop deblocking filter is applied to remove blocking artifacts which can improve both objective and subjective quality and also improve the inter prediction; arithmetic and context-adaptive entropy coding is included. For further description of H.264 it is referred to [8] [9].

Similar to prior standards, the concept of profiles and levels is used in H.264. These define subsets of available coding tools and constraints on key parameters to enable efficient implementations and interoperability for a wide range of applications [7]. Three profiles were originally defined in the first version of the standard: Baseline, Main and Extended. The Baseline profile is suited for conversational services and supports some low-delay error resilience and coding tools of relatively low complexity. For instance, it does not allow B-slices. The Main profile is defined to allow tools that maximizes coding efficiency, which makes it a good choice for typical video entertainment applications such as television broadcasting. The Extended profile is a superset of Baseline and include tools to support coding efficiency, error resilience and resynchronization. Streaming services would probably utilize this or the Baseline profile. The Baseline profile is assumed for the remaining text as this profile is most suitable for real-time communication systems.

2.1.2 H.264 over IP

As already mentioned, one of the main goals for JVT when developing H.264 was to improve the network friendliness. In order to achieve better flexibility and customizability the design is divided into a Video Coding Layer (VCL) and a Network Abstraction Layer (NAL), as illustrated in Figure 2.2. The VCL, which is where the architecture in Figure 2.1 is located, consists of the core compression algorithms and is designed to be as network independent as possible. The NAL

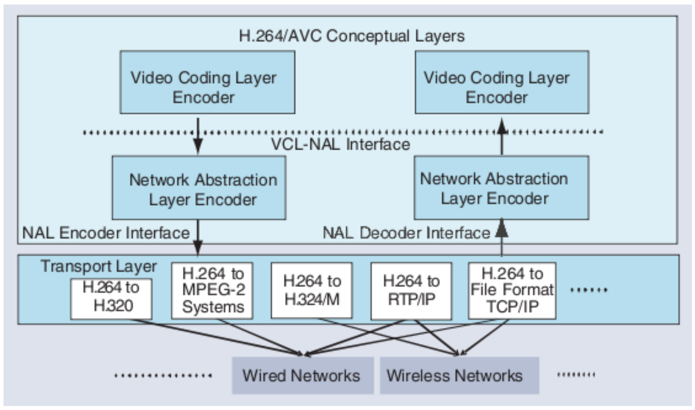

H.264/AVC Conceptual Layers
Video Coding Layer Encoder
Video Coding Layer Encoder
VCL-NAL Interface
Network Abstraction Layer Encoder
Network Abstraction Layer Encoder
NAL Encoder Interface
NAL Decoder Interface
Transport Layer
H.264 to H.320
H.264 to MPEG-2 Systems
H264 to H.324/M
H264 to RTP/IP
H.264 to File Format TCP/IP
Wired Networks
Wireless Networks

Figure 2.2: The H.264 standard in transport environment [9].

on the other hand, maps the VCL data to different transport layers [10]. In this section, properties of H.264 when it is transmitted over Internet Protocol (IP) are studied.

The Internet Protocol is a connectionless network protocol, where the sender can not guarantee that there will be no loss, error insertion, misdelivery, duplication or out-of-sequence delivery of a packet. IP networks typically employ two different transport protocols: User Datagram Protocol (UDP) and Transmission Control Protocol (TCP). The latter is a connection-oriented protocol which guarantees reliable and in-order delivery of packets from sender to receiver. This is accomplished by re-transmissions and timeout mechanisms. Consequently, there is no guaranteed upper bound for delay and jitter. For services that decode the media while it is being delivered over the network, it is critical to keep the delay and jitter below an acceptable threshold [11]. Hence, UDP is normally employed since it is faster and often more efficient than TCP. It is preferred despite the fact that it offers only the same best-effort services as IP,

Real-time Transport Protocol (RTP) is usually utilized on top of UDP, together with RTCP. RTP carries the real-time data while RTCP monitors the Quality of Service (QoS). The RTP-header contains information, e.g. sequence number and timestamp, which enables the receiver to remove duplicate packets, put the packets in correct order and detect packet losses. Packets which arrive too late or not at all are considered lost [10]. By this, the receiver is able to measure the QoS and report back to the sender using RTCP in order to adjust the encoder according to the network conditions.

To summarize, IP/UDP/RTP is the most common protocol combination for multimedia systems that consume the media while it is being delivered over the network. No QoS guarantees are given. However, the encoder is able to receive feedback in order to improve the QoS.

2.2 An introduction to error robustness

In a best-effort network environment as described in previous section, it is important to minimize the effect of errors when they occur. First and foremost, the encoded video must still be decodable. Second, the visual quality should be as good as possible. Many approaches exist that aim to improve the robustness against loss and corruption of data in a video stream. In this section we divide such techniques into three main categories: error resilient schemes applied in the encoder, error concealment at the decoder side, and techniques that require interaction between the decoder and encoder through a feedback channel. We will describe some of the most common techniques within each category.

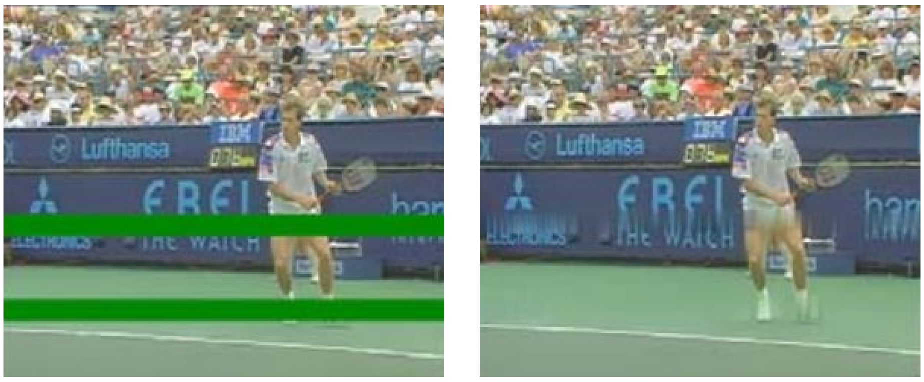
2.2.1 Error resilience

Error resilient encoding operates independently of the decoder. The idea is to introduce extra information in the bit stream, also known as redundancy. When there are transmission errors this additional information helps the decoder to conceal and recover from the errors, hopefully without unacceptable distortion. The design goal for such encoders is to achieve a maximum gain in error robustness with the smallest amount of redundancy. However, compared to coders optimized for coding efficiency, error resilient encoders typically are less efficient. In other words, they use more bits to obtain the same video quality in the absence of transmission errors.

Intra coded MBs is one of the most obvious and powerful error resilience tools. It is performed either for a single MB or at slice or frame level. Historically, intra coded MBs resynchronize the video for the part of the frame it is applied, since it does not depend on previous frames. In H.264 however, this is only partly true as intra prediction is allowed to be performed from inter predicted MBs. This is good for coding efficiency, but there is a higher risk for errors to propagate to neighboring blocks within the same frame. In order to improve the resynchronization property for intra MBs, it is possible to use *constrained intra prediction* [8]. This prevents intra MBs from being intra predicted from inter MBs. Naturally, this increases the error robustness but decreases the coding efficiency. Nevertheless, [12] suggests that constrained intra prediction should always be used in erroneous environments because the quality degradation in error-free environments are only small. The full resynchronization property is achieved by applying an *Instantaneous Decoding Refresh (IDR).* This is an intra-frame, but in contrast to regular intra-frames, an IDR indicates that that no subsequent pictures in the stream will require reference to pictures prior to the IDR.

It is important to remember that intra coded MBs use significantly more bits than those which are inter predicted. For instance, P-slices are approximately only 10% the size of I-slices [13]. Hence, it is important to not overdo the number of intra coded MBs.

Slice mode is another resilience tool which forces a slice to contain a given number of MBs or bytes. As described in Section 2.1.1, intra prediction can only be carried out inside the slice segment. Thus, slice structuring avoids error propagation across slice borders within the same frame. Forcing a higher number of slices within a frame therefore reduces the consequence of erroneous or lost data in a slice. In addition, if the slice size exceeds the Maximum Transmission Unit (MTU) of the network, the slice is divided into several fragments (fragmentation). If one fragment is lost during transport, the complete slice must be discarded. Also, the error probability in packet-switched best-effort networks normally decreases when the packet-size gets smaller, which is possible when decreasing the slice size. As before, this degrades coding efficiency due



(a) Corrupted frame (b) Recovered frame

Figure 2.3: Recovered intra-frame using a hybrid error concealment strategy [15].

to less efficient intra prediction and because more slice information needs to be transmitted.

2.2.2 Error concealment

Error resilience techniques alone is normally not enough to retain an acceptable quality. It is also necessary to minimize the effect errors on the decoder side. The process of recovering and estimating the lost information due to error transmission is referred to as error concealment. It is not a part of the H.264 standard and many concealment algorithms exist. This thesis does not cover evaluation of different concealment schemes. Thus, this section will only describe the main concepts behind this group of robustness techniques.

Errors are detected by a violation in the defined syntax and/or semantics. Upon detection, the missing or corrupted packets are simply discarded and a robust decoder must resynchronize the stream and conceal the corruption [14]. The basic idea behind error concealment is to estimate lost portions of the picture from correctly received data. This can be achieved with temporal, spatial or hybrid techniques where some sort of interpolation usually is applied. With a temporal strategy, the values of a damaged MB is typically computed by interpolating pixels from a previously decoded frame. Often these schemes also use available or estimated MVs to improve the concealment. Spatial interpolation employs only pixels from neighboring areas in the same frame to recover from the error. Various interpolation algorithms are available for both temporal and spatial concealment and may have a large impact on the result. Hybrid techniques are a combination of the previous ones. Figure 2.3 illustrates the effect of adopting error concealment.

More complex schemes usually give better quality, but also higher processing time. Hybrid schemes, for instance, are normally slower than techniques using temporal replacement, but yield better results. In many systems, such as real-time communication systems, processing time is a critical factor and suboptimal solutions are preferred [15].

2.2.3 Feedback-based error control

Error resilience and concealment as presented thus far, let the encoder and decoder operate independently when it comes to combating transmission errors. Feedback-based error control, on the other hand, requires the encoder and decoder to interact. If a feedback channel is available, the decoder can inform the encoder about detected errors and the encoder can adapt its operations accordingly to reduce or even eliminate the effect of these errors. There are several aspects by such error control that need to be considered, such as the timing rules of the feedback, what information the decoder should provide the encoder, and how the encoder should adjust its encoding process. In this section we will introduce the principles behind feedback-based error control. Section 2.3 provides a more detailed presentation of the most interesting feedback mechanisms for video communications today.

The feedback messages contain information useful for both proactive actions to prevent future transmission errors and/or reactive actions to recover from reported corruption.[2] An example of the former is to reduce the video bit rate if network congestion is reported. This thesis, however, focuses on the latter. For instance, the decoder may provide ACKnowledgements (ACKs) or Negative ACKnowledgements (NACKs) about received data, either at transport or application level. That is, either report packets or specific parts of the video sequence as received or lost. Upon such feedback, the encoder acts to recover from the damage that has occurred.

There are several ways the encoder could employ this information to improve the error robustness. The simplest approach is to resend the lost packets (Automatic Repeat reQuest (ARQ)), but this is not an option for low-delay video systems because of the timing requirements. Instead, the encoder accepts that some data have been lost, and adapts its operation to recover from the errors as soon as possible. Traditionally, there are two main approaches to do this: The current frame can be encoded by referencing only to previous frames that are successfully received. This approach is referred to as Reference Picture Selection (RPS) [16]. Or, the current frame may intra code the corrupted region in order to stop the error propagation, commonly referred to as intra refresh [16].

RPS may either use ACK or NACK. If ACK is applied, the encoder *always* use reference frames that are confirmed successfully received. This approach is very robust as it entirely avoids error propagation. However, it lowers the coding efficiency since the motion prediction becomes less efficient when older reference frames are utilized. In addition, more bandwidth is used for feedback reports compared with NACK. Hence, it is more common to apply NACK with RPS. With NACK, the encoder operates as normal until it learns about damaged parts in a previously coded frame. Then, even older frames than the damaged one, are used as reference because these are assumed to have been received correctly. Using this approach, transmission errors usually become visible in the decoded video, but they will be corrected after a short while. The advantage is that the coding efficiency is improved compared with using ACK, since newer reference frames are applied as long as no errors are reported.

The second main approach is to intra refresh the corrupted regions. NACKs are used to report losses to the encoder. By using stored MVs of previously

[2]Proactive actions are operations performed to minimize the effect of a possible future loss. Reactive actions are performed after an error is registered.

coded frames, the encoder tracks how these losses have affected the decoded video. Regions of the current frame that are subject to corruption are identified. MBs that belong to these regions are forced to be intra coded in the current frame. Hence, if the tracking is correct, all errors are repaired. The intra refresh may lead to severely reduced coding efficiency of the current frame. For other frames, however, the coding efficiency is unaffected. More information on feedback-based error control may be found in [16].

The performance of a feedback-based ER scheme depends highly on the timing rules of the feedback mechanism. Timely arrival of feedback information is crucial. If the encoder receives the feedback too late, it may result in an increased temporal and spatial error propagation, which degrades the perceptual quality. On the other hand, if all detected errors are to be reported instantly, the feedback channel would require a large bandwidth if the error rate is high. This may lower the bandwidth available for the video stream, which also leads to decreased video quality. Obviously, it is desirable with feedback transmission rules that balance these factors well.

2.3 Feedback mechanisms for H.264 over RTP

This section describes the most important standards for providing feedback useful for error control in real-time low-delay video applications. These standards define the content of the feedback and/or its temporal characteristics.

2.3.1 RTP/RTCP

As explained in Section 2.1.2 RTP/RTCP [5] is the most common protocol to transport time critical audio and video over the Internet today. RTCP provides a way for the encoder and decoder to exchange out-of-band control information for the RTP flow, in both directions. Several packet types are available, but RTCP's primary function, that is to provide feedback on the overall reception quality, is carried out through Sender Reports (SRs) and Receiver Reports (RRs). Network characteristics such as packet loss ratio, round-trip delay and jitter are made available by these reports. Utilizing this knowledge the encoder can adapt its coding scheme and transmission behaviour to the observed network quality. For instance, in the case of a high packet loss ratio the encoder should apply some ER scheme to make the video stream more robust to errors.

To ensure a good balance between the bandwidth used for media data and feedback it is recommended that an additional 5% of the session bandwidth is used for RTCP packets [5]. It is also recommended a fixed minimum transmission interval of five seconds between each RTCP packet to avoid feedback bursts in special situations. In order to obtain more accurate statistics, SRs and RRs should be sent as often as allowed within these constraints in a near periodic fashion.

2.3.2 RTP/AVPF

The Extended RTP Profile for RTCP-based Feedback (RTP/AVPF) [4] specifies modifications and additions to RTCP. It enables more immediate feedback to the senders without violating the RTCP bandwidth constraint. Furthermore, it has the ability to provide more useful information in terms of error control, e.g. *what* data are missing. This allows the encoder to implement short-term adaption techniques and efficient repair mechanisms.

To achieve timely feedback suitable for error recovery, the report interval in [5] is changed and a new RTCP packet type is added, namely the FeedBack (FB) message. This message may, for instance, indicate loss or reception of particular pieces of a video stream. FB messages are classified in three categories: transport layer FB messages, payload-specific FB messages and application layer FB messages. The FB message is only a new RTCP packet type and is transmitted as a compound RTCP packet together with other RTCP messages, such as the RR. Hence, both reception statistics and information for immediate repair of a video stream is provided to the sender.

The transport layer FB messages are independent of the particular codec or application in use. RTP/AVPF currently defines only one such message, a generic NACK, that may be used to report which RTP packets the receiver did not receive. The ER scheme described in Section 2.5 applies this message.

Payload-specific FB messages may provide more detailed feedback since these are tailored the video stream payload types. They may for instance be used to indicate loss or corruption of one or several MBs in scan order, or report which pictures may safely be used as reference pictures (if multiple reference pictures is employed in the encoder).

Application layer FB messages is used to transport application-defined data directly from the receiver's to the sender's application. The content of these are not defined by [4], except for the message format to identify such messages.

The various FB messages described above are of most use if they are sent early after an error is detected. RTP/AVPF defines a modified RTCP transmission algorithm that allows immediate feedback as long as the RTCP bandwidth constraint is not exceeded. The standard [4] outlines the general behaviour as follows:

> As long as no FB messages have to be conveyed, compound RTCP packets are sent following the rules of RTP, except that the five second minimum interval between RTCP reports is not enforced. Hence, the interval between RTCP reports is only derived from the average RTCP packet size and the RTCP bandwidth share available to the RTP/RTCP entity.
>
> If a receiver detects the need to send an FB message, it may do so earlier than the next regular RTCP reporting interval. (...) It checks whether it is allowed to send Early feedback. If sending Early feedback is permissible, the receiver sends the FB message as part of a minimal compound RTCP packet. The permission to send Early feedback depends on the type of the previous RTCP packet sent by this receiver and the time the previous Early feedback message was sent.

These rules become more understandable by looking at a typical scenario illustrated in Figure 2.4. Various events may be defined to generate the need of FB messages, but in this scenario we use the loss of one or several RTP packets as such an incident.

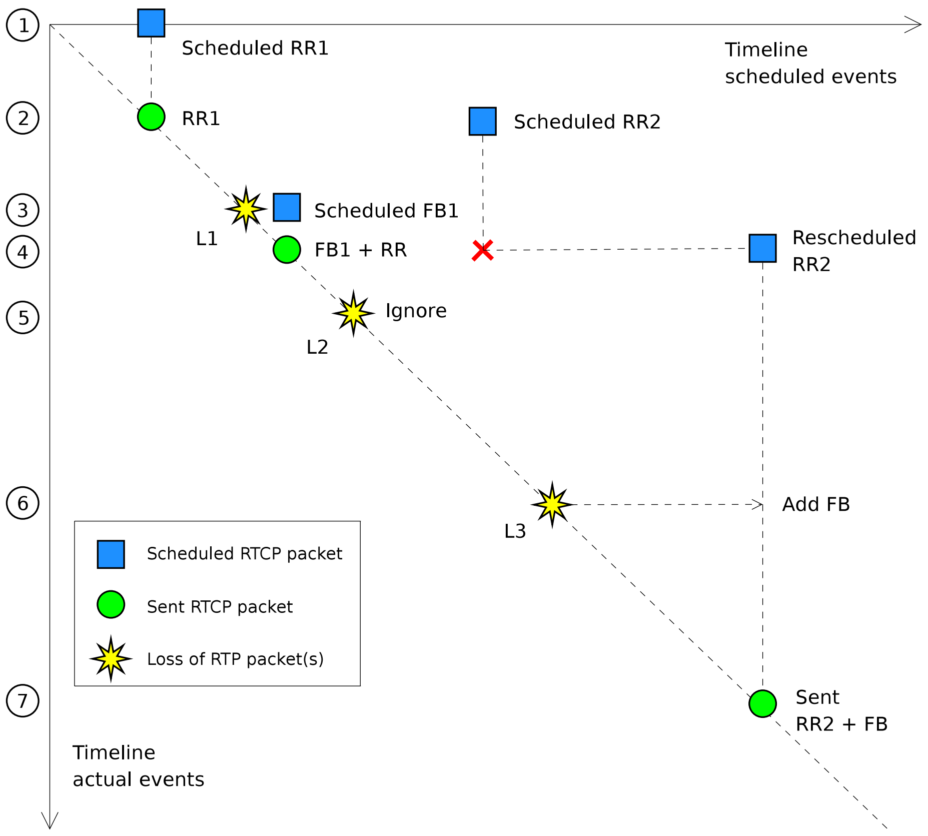


Figure 2.4: RTP/AVPF scenario. The time for the occurrence of events/actions increase vertically downwards. The horizontal axis illustrate when a scheduled action is planned to be executed. The figure should be read by going vertically down through points 1 to 7. Events/actions that are on the diagonal dotted line occur/are executed at this time. Actions to the right of this line are scheduled to a future time.

1. At the beginning of the session a regular RTCP packet is scheduled for transmission following the rules of RTP/RTCP, except that the five second minimum interval between RTCP reports is not enforced.

2. As long as no FB messages have to be conveyed before the scheduled time, i.e. no loss needs to be reported, the regular RTCP report scheduled at point 1 is sent as normal. Then a new RTCP packet is scheduled for transmission according to the same algorithm. The interval is derived from the average RTCP packet size and the RTCP bandwidth share available to the RTP/RTCP entity (except for a randomization factor).

3. If the receiver experience a loss before the next regular RTCP report is transmitted, an FB message is scheduled for transmission early after the incident. The interval is random, but small, and its upper limit is bound by an application-specific parameter.

4. The FB message is sent at the scheduled time. Besides the feedback information, other RTCP messages are added to the RTCP packet, such as an RR. Because of the bandwidth constraint, the next scheduled regular RTCP packet is rescheduled so that the transmission interval is doubled (except for a randomization factor). Until this time is reached the receiver is not allowed to send any FB messages or other types of RTCP packets.

5. If losses occur while Early feedback is forbidden, the receiver may choose to add an FB message to the next scheduled regular RTCP report if it is transmitted soon. This decision is made by an application-specific parameter that defines the upper bound for an FB message to an event to be useful. If this upper bound is exceeded, the loss is simply ignored. This may, for instance, cause the encoder to not being able to recover from the error.

6. If the upper bound mentioned in point 5 is not exceeded, an FB message is added to the next regular RTCP report. It is allowed to report several lost packets in one RTCP packet. Thus, if further losses occur before the transmission of this RTCP packet, additional FB messages may be added.

7. When the scheduled time for the regular RTCP report is reached, the RTCP packet is sent containing an RR (and other RTCP message types) and the added FB messages if any. From this time the receiver is allowed to send Early FB messages again, and a new regular RTCP report is scheduled using the same transmission interval algorithm as in point 1 and 2. The state of the receiver is now the same as in point 2, i.e. if a new loss occur and Early feedback may be sent.

As already mentioned, FB messages may contain other kind of information than lost RTP packets. Besides the already described message types, there is a work in progress aiming to provide more possible feedback information. The IETF-draft Codec Control in the RTP Audio-Visual Profile with Feedback (AVPF/CCM) [17] adds new FB message types, both payload-specific and transport layer FB messages. Among the new features are the ability for the receiver to request a temporary maximum media bit-rate; to request a temporal spatial trade-off; to request a full intra refresh; and to send a Video Back-Channel Message (VBCM) as described in the next section.

2.3.3 H.271

H.271 [18] specifies the format of VBCMs for conveyance of status information and requests from a video receiver to a video sender. The syntax is applicable for use with the majority of the existing video coding standards, e.g. H.261, H.263 and H.264. There exists some overlap between H.271 messages and the messages defined by RTP/AVPF and AVPF/CCM. For instance, reporting the loss of MBs and/or pictures is possible using both VBCM as defined in H.271 and the "native" FB messages [17]. But there are some differences, such as H.271 being able to provide feedback about received parameter sets.

H.271 does not discuss any timing rules for the VBCMs. But as mentioned in the previous section there is a draft [17] that allows VBCMs to be sent as FB messages using RTP/AVPF. The timing rules described in previous section must then be followed since the VBCM is nothing more but a payload-specific FB message. Applying H.271 together with RTP/AVPF then extends the possibilities of the feedback.

2.3.4 H.245

H.245 [19] is the control protocol for H.323-based systems[3]. It is used to exchange end-to-end control messages governing the operation of the H.323 endpoints. These control messages carry information related to exchange of capabilities, opening and closing of logical channels used to carry media streams, flow-control, and general commands and indications. The latter includes messages that may be used in QoS management. For instance, the spatial and temporal location of erroneous MBs in the decoded video may be reported by sending commands to the encoder, such as `VideoFastUpdatePicture`, `VideoFastUpdateGOB` or `VideoFastUpdateMB`. The reader is referred to [21] for a full list of commands used for QoS management.

The decoder may send the commands above in order to recover from errors. However, this is not the primary function of H.245. Moreover, it may only be used in H.323-based systems. Hence, in practice, H.245 is only an alternative to a limited set of applications. Thus, other protocols are preferred in order to develop a general feedback mechanism. In this regard, it should be noticed that H.323 transmits media using RTP, which implies that both RTCP and RTP/AVPF may be used in conjunction with H.245 for H.323-based systems.

2.4 Related works

Extensive research has been done in the field of error robustness for the latest video coding standards. Wang et al. [22] give an introduction to robustness techniques used for real-time video communication systems. They present general robustness principles for both the encoder and decoder as well as feedback-based techniques.

Kumar et al. [14] give a review of available robustness tools in H.264, and summarize some of the experimental results obtained in other researches. Wenger [10] simulates the performance of some error resilient encoding techniques applicable for low-delay applications over best-effort IP networks.

A review of feedback-based error control approaches are found in [16]. Although it was written with H.263 in mind, the techniques are generally valid for H.264. A comparison of two feedback schemes, one using intra refresh and the other RPS, is illustrated. It is shown that RPS performs better for high error rates, while for low error probability the situation is reversed.

Several works have presented novel feedback-based ER methods. The least complex scheme is probably adaptive intra-frame refreshment [23]. It reports lost packets to the encoder, which codes the next frame in intra mode and decreases the intra frame refreshment interval.

In [24], three RPS schemes are presented, all which use flexible reference frames. This means that the encoder use only *one* frame that it knows has been

[3]H.323 [20] is an umbrella recommendation that specifies the components, protocols and procedures to provide multimedia communication services (real-time audio, video, and data communications) over packet networks.

received correctly as reference.

Yu et al. [25] developed another RPS scheme utilizing multiframe encoding. In this scheme, several frames may be used as reference for a frame. When an error is reported, only frames that are known to have been received correctly are used as reference for the next frames.

The schemes referred to above are all interesting approaches on how to employ feedback information to improve the error robustness. However, they all assume that feedback, primarily ACK and/or NACK, always is sent instantly from the decoder. There are two main problems with this. First, in real-world applications there are bandwidth limitations. If the feedback always shall be transmitted instantly, the feedback mechanism can occupy a too high ratio of the available bandwidth, which may cause a degrade in video quality. Second, a standardized feedback scheme is necessary for interoperability between applications. Such a standard may provide other features than those assumed in the above-mentioned studies. Hence, it is important to examine how feedback-based error control schemes perform in realistic environments using standardized feedback mechanism.

2.5 A feedback-based error control scheme

The complete feedback-based ER scheme developed and employed in this thesis is outlined in this section. The scheme utilizes RTP/AVPF as the feedback protocol to report lost parts of the video stream. The encoder performs error tracking using this information and eliminates the corruption by intra refreshing the erroneous areas. The scheme is referred to as Feedback-Based Intra Refresh (FBIR).

2.5.1 Reporting errors

RTP/AVPF in NACK mode is considered to provide adequate feedback information for this scheme. If the encoder forces a fixed number of MBs in each slice ("slice mode"), and one slice in each RTP packet, it is easy to identify lost portions of the video given the sequence numbers of the lost RTP packets. The non-fixed parameters that affect the timing intervals are set to a trade-off between instant feedback and the ability to report possible burst losses. The actual interval will depend on available RTCP bandwidth, the number of participants etc., and is examined in Section 4.3 and 5.1.

2.5.2 Tracking the corrupted area

After the lost portions of the video are known, the encoder is able to calculate the affected areas in the decoded video. This process is known as error tracking and is necessary for any feedback-based ER scheme that utilizes intra refresh as the recovery tool. Without perfect error tracking some errors may still persist after the intra refresh has been applied and propagate further into the video. That is, unless full intra refresh of the whole picture is used.

Error tracking is not a trivial task and may require too much processing power for use in real-time low-delay video applications. Thus, our scheme employs an approximation of Chang and Lee's fast error tracking algorithm [6].

Their fast algorithm is a compromise between precision and speed, hence only near perfect error tracking is achieved. The fast algorithm is derived from a precise error tracking scheme [6], which may be outlined as follows: When the encoder becomes aware of one or more corrupted MBs in a prior frame, it loops over all MBs in the current frame and computes a Contamination Ratio (CR) for each MB. The CR reflects how many corrupted pixels there may be in that MB. In order to decide if a pixel is corrupted, stored MVs are used to trace back the temporal dependencies of the pixel in question. If the value of the pixel is (in)directly predicted from a corrupted area, this pixel is also considered corrupted. Based on the CR for the current MB, the encoder decides if it should force it to be intra coded in order to recover from the error. This algorithm is not able to track propagation caused by intra prediction. However, this is not a problem since intra prediction is allowed only within one slice, as explained in Section 2.1.1, and an integer number of slices are corrupted/lost. Thus, the algorithm yields precise error tracking.

Applying the algorithm outlined above increases the memory requirements of the encoder since the MVs of a fixed number of previous frames have to be stored. It also requires more processing power than what may be desirable. For a comprehensive description of these requirements the reader is referred to [6]. In order to reduce the computation requirements a “four-corner tracking approximation” is applied [6]. The backward motion dependencies are first examined for only four corners of an MB, instead of for each pixel. If any of the corners are contaminated, precise error tracking is performed on this MB to obtain its CR. The error tracking is no longer precise, but the computation complexity is reduced by 64 folds without significant loss in precision.

2.5.3 Selective intra refresh

After the corrupted area has been disclosed, the encoder performs an intra refresh of the affected MBs in order to eliminate the errors. But not all errors are equally important to correct. For instance, corrupted MBs with very low CR may not cause visible distortion. Thus, MBs with higher CR should be prioritized in the recovery process. A more important case is when the lost information is perfectly concealed by the decoder using techniques described in Section 2.2.2. Then it is unnecessary to waste bits on intra refreshing that area. Most temporal and hybrid concealment schemes conceal the parts of the video where there is no motion at the time of the error perfectly.

In order to improve the performance of the FBIR scheme, a simple algorithm for detecting static regions is developed and applied in the encoder. This detects MBs that are unnecessary to correct, and thus reduces the number of bits used in the recovery process. The procedure is shown in Algorithm 1 and can be summarized as follows: For each MB in the picture, the last frame for which this location was *not* considered “near static” is recorded. The “near static” condition depends on the coding mode and MVs of the MB: If the MB is coded in skip mode or its MVs is smaller than a defined threshold T, the MB is classified as “near static”. This operation is performed for every encoded MB. The condition includes small MVs because camera noise or similar effects may cause the encoder to calculate negligible MVs, even if the area is visually static. Such areas are also concealed very well and do not need to be refreshed. When an error is reported and the error tracking indicates that an MB should be intra refreshed, the encoder checks the "near static" history of this location. If the location has been classified as "near static" for all frames since the error occurred, the intra refresh is suppressed. This algorithm suppresses intra coding of regions that are concealed near perfectly by the decoder, while still refreshing regions corrupted by loss or error propagation. Hence, the coding efficiency is improved if the video contains such static areas.

Algorithm 1 Detection of "near static" macroblocks

```
/* Execute for each coded frame */
for all MBs $i$ in frame $N$ do
    if ($mode_i \neq$ SKIP) and (ABS($mv_i$) $> T$) then
        $last\_non\_nearstatic_i \leftarrow N$
    end if
end for

/* When errors occur, if $last\_non\_nearstatic_i < error\_frame$, */
/* do not force intra refresh. */
```

2.5.4 Additional robustness tools

Besides FBIR described in the previous sections, a couple of other robustness tools (described in Section 2.2.1) are applied. First, "slice mode" is enabled so that each slice contains exactly one Group Of Blocks (GOB) (one "row" of MBs). This is to enable identification of the lost MBs based on the RTP sequence numbers. Other slice sizes can also be applied, as long as there is a fixed number of MBs in the slice. Second, the number of reference frames is set to one, i.e the encoder is only allowed to perform intra prediction from the previous frame. This is necessary in order to prevent reference to a corrupted region after intra refresh is performed. Third, constrained intra prediction is used as recommended by [12] to increase the resynchronization property of the intra coded MBs. If not applied, an intra coded MB could in theory be intra predicted from an erroneous inter predicted MB in the same slice.

2.6 Intelligent Packet Loss Recovery

This section describes a simplified version of TANDBERG's[4] solution for error robustness in H.264. It is designed for use in video communication systems at relatively high packet loss rates. The scheme is called Intelligent Packet Loss Recovery (IPLR).

IPLR [2] [3] is a proactive error resilience scheme. It forces intra refresh of MBs according to a motion-based routine, which spreads the intra MBs over several frames. The scheme is outlined in Algorithm 2. A counter is assigned to each MB and is updated based on the MB's coding mode for every frame. The mode of the MB at the same location in the next frame may be overruled forced to intra based on the value of this counter. MBs which often are in skip

[4]TANDBERG is one of the world's largest manufacturers of video conferencing systems for the business market.

Algorithm 2 IPLR

```
/* Execute for each coded frame */
for all MBs $i$ in frame $N$ do
    if $mode_i$ = SKIP and $mv_i = 0$ then
        $counter_i \leftarrow -10$
    else if $mode_i$ = INTER then
        $counter_i \leftarrow counter_i + 1$
    else
        $counter_i \leftarrow 0$
    end if
    if $counter_i > 1$ then
        ForceIntraMB($N + 1$, $i$)
        $counter_i \leftarrow 0$
    end if
end for
```

mode and have motion vectors set to null are considered static (and are usually a part of a static background). These will not be intra updated as often as other parts of the frame, if at all. Conversely, parts of the frame with high motion activity which normally are coded as inter MBs, will more often be subject to intra refresh. Hence, error robustness is only introduced in the parts of the frame where there is motion. IPLR is also compliant with the H.261 and H.263 standards [2].

2.7 The H.264 reference software

The H.264/AVC reference software [26], commonly referred to as JM, is developed to "aid users of the video coding standard to establish and test conformance and interoperability, and to educate users and demonstrate the capabilities of the standard" [27]. Its source code is available for free [26]. It implements the normative decoding process defined by the standard, but also a flexible encoder and non-normative error concealment techniques. The encoding/decoding process is configurable through a configuration file [28]. Several options for increased error resilience are accessible, such as the intra-frame period, random intra refresh of MBs, constrained intra prediction, slice structuring and more.

The JM decoder implements both a temporal and spatial error concealment scheme [27]. The temporal scheme is applied when parts of an inter-frame is missing. It tries to estimate the MVs for the missing MBs based on the MVs of its successfully decoded neighboring MBs. It uses this estimation and information from already decoded frames to conceal the error. For missing parts of an intra frame, the missing MBs are spatially predicted from neighboring MBs using weighted sample averaging. If a whole frame is missing, two options are available for the user; either copy the entire previous frame (frame copy), or copy its motion vectors and predict the missing frame (motion copy).

2.8 Measures for video quality

When analyzing different video sequences it is necessary to produce some quantifiable results which can be used for comparison. This section outlines the idea behind subjective testing and explains more in detail the most widespread objective measure: Peak Signal-to-Noise Ratio (PSNR).

Subjective quality assessments are generally generated by having many viewers (preferably experts) evaluate several test sequences. They rate the videos subjectively according to some predefined rating system. These ratings are then used for statistical calculations of the quality. Assessments using this approach achieve good results regarding how the quality is perceived by humans. However, this process is not only time-consuming but also tedious and expensive to perform. Hence, objective measures are often preferred.

Objective assessments are usually easier and faster to apply. However, they do not coincide with perceptual quality as well as subjective techniques do. This is because it is hard to develop algorithms that imitate the properties of the Human Visual System (HVS). This is also the case for PSNR. PSNR may be used for both images and video sequences. It derives the distortion between the original and reconstructed image/video pixel by pixel. A higher PSNR value corresponds to better quality. For a monochromatic video sequence of T frames and pictures of size MxN, the measure is defined as [29]

$$\mathrm{PSNR}_{\mathrm{dB}} = 10\log\frac{(2^b-1)^2}{\frac{1}{TMN}\sum_t\sum_m\sum_n[\mathrm{I}(t,m,n)-\mathrm{K}(t,m,n)]^2}, \tag{2.1}$$

where either I or K is the original sequence and the other one is the impaired sequence, and b is the number of bits used to represent a pixel value (and is normally 8). For colored images/videos it is common to calculate the PSNR of the luminance component only, denoted Y-PSNR, as this is the most important component for the HVS.

A great deal of effort has been made to develop objective quality assessments that incorporates perceptual quality measures by considering HVS characteristics [29]. However, these methods are far more complex than PSNR and are not widely used yet.

CHAPTER 3

Methodology

This section deals with the methodology of the study. It provides an outline of the research and planning process, the implementation and necessary simplifications, the simulation set-up and approach for evaluation of the ER schemes. Some of the methods used may not be optimal regarding quality and reliability. However, after careful consideration of the trade-off between complexity, duration and accuracy, the chosen methods were preferred.

The study can be divided into four main parts: research on related topics and planning of the following implementation and simulation process; implementation of the feedback mechanism and the corresponding ER scheme; online simulations schemes utilizing a network emulator; and finally, measurements and evaluation of the decoded video quality.

3.1 Research and planning

The research performed in [3] served as a ground for some of the topics covered in this study. These topics were H.264, proactive error resilience including IPLR, error concealment, and network characteristics of real-time multimedia traffic. The research performed in this study covered mainly the topics on various feedback mechanisms and how to employ this information in the encoder. This included an in-depth study of the RTP/RTCP and RTP/AVPF standards and understanding the alternative feedback schemes. Prior research on feedback-based error control was important to decide how the encoding process could utilize the available feedback information. Information was gathered through reading scientific papers, relevant standards, and discussing with employees at Q2S[1].

[1]Q2S - The Centre for Quantifiable Quality of Service in Communication Systems [30] is a Norwegian Centre of Excellence at The Norwegian University of Science and Technology in Trondheim.

3.2 Implementation

JM 11.0 [26] was used for encoding and decoding of the video sequences. The previously modified encoder with IPLR support [3] was employed. JM does not initially support real-time encoding and network transmission, so this feature was also added. Only the necessary features of RTP was implemented for the encoder to be able to send the coded bit stream to one single decoder in real-time. The RTP implementation allowed packets to be out of sequence and duplicated, but if they were 50 ms later than expected they were considered lost. As described in Section 2.5, RTP/AVPF was selected as the feedback protocol. All mandatory parts of this standard, such as RRs and Source DEScription (SDES) with CNAME items, was implemented in addition to the NACK FB messages. Implementation-specific constants were set to wait for detection of loss bursts while preserving a relatively instant feedback. The maximum time to hold an Early feedback was set to 150 ms. Appendix C lists other implementation-specific constants and their values.

The error tracking and selective intra refresh was implemented according to the algorithms described in Section 2.5 except from one approximation. In the JM encoder, the decision to force an MB to be intra coded is made before the MVs are calculated. This implies that the MVs for the current MB were not available at the time error tracking and intra mode selection was performed. Hence, a prediction[2] of these MVs was used instead. (But exact values of earlier MVs was of course available and used.)

For more details regarding the implementation the reader is referred to Appendix C.

3.3 Simulation

3.3.1 Testing environment

Online simulations were needed for the encoder to adapt its procedure according the feedback provided by the decoder. The test bed for IP networks described by Hillestad et. al. [31] was redesigned to allow both online encoding and decoding. Figure 3.1 shows the resulting test bed setup. The encoder sent the compressed video as an RTP stream, via a network emulator and a packet capture device, to the decoder. The PacketSphere Network Emulator by Empirix [32] was employed. It imitates the behaviour of a real-world network and is configurable through several network attributes. Only the loss rate and network delay was altered in our tests. The resulting data stream was sent through a network monitoring interface card. This device saved the captured packets to file to enable offline inspection of the packet flow at a later time. An Endace DAG3.5E card [33] was used. Finally, the encoded video arrived at the decoder and was processed in real-time. The received video was saved to file for measuring and visual inspection at a later time. While decoding, the decoder generated feedback according to the RTP/AVPF standard. These packets were transmitted to the encoder using the reverse route, except that no loss were introduced by the emulator. This was to reduce the number of free variables in the simulations and

[2]The prediction algorithm was already implemented in JM for use in the encoding process.

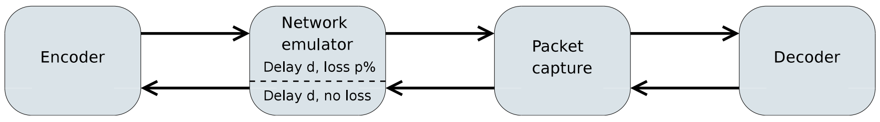


Figure 3.1: Test bed overview.

because there are means to ensure near error free transmission of RTCP[3]. The network elements were connected in a 100 MBit/s Local Area Network (LAN), so the amount of uncontrolled loss and delay was negligible.

3.3.2 Sequences and parameter sets

Three video sequences were used for testing: *Conversation* contains little motion, has a static background and describes a typical video conference setting; *Foreman* is widely used and has a medium motion level; and *Soccer* has a high motion level. The H.264 reference software is not optimized with respect to computation efficiency and is very slow. Hence, the sequences had to be converted to QCIF format and 10 frames per second (fps) to be able to encode them in real-time. This conversion was performed with the tools AviSynth [34] and VirtualDub [35]. The "Common Conditions for wire-line, low delay IP/UDP/RTP packet loss resilient testing" [36] use sequences of minimum 4000 frames to avoid the influence of distribution errors in its error patterns. Thus, our test sequences were also extended to 4000 frames by joining them with reversed versions of themselves in a loop. This approach did not introduce any scene cuts into the video clip, which is desirable for realistic simulations since there usually are no cuts in the video sent from one endpoint in a conversational system.

The simulation parameter sets were chosen to reflect a wide set of realistic network environments, but also to test the ER schemes in more extreme conditions. Loss rates of 0, 1, 3, 5% with bursts of 1–3 packets were chosen, partly based on information from [37]. The network delay was set to be fixed to reduce the number of free variables. Two delays were tested: 50 ms and 200 ms. 50 ms because it has been shown to be a typical network delay over long distances [38], while 200 ms has been defined to be the maximum tolerable delay in interactive multimedia applications [11]. The RTP session bandwidth were set to 144 and 64 kbit/s according to [36]. In addition, simulations were performed at 384 kbit/s to compare when the distortion introduced by the encoder was negligible. These bit rates include the IP/UDP/RTP-headers, which are 40 bytes per packet. Hence, the source coding bit rates, excluding the packet overhead, are 28.8 kbit/s lower than the listed rates.[4]

The encoder configuration parameters were set up according to the baseline profile. The parameters were set to improve the coding speed at the cost of reduced coding efficiency to manage real-time encoding. Appendix B.1 summarizes the most important parameters. In order to keep the bit rate at approximately the defined RTP session bandwidth, the quantization parameters were adjusted for each ER scheme and parameter set. These quantization parameters

[3]Near error free transmission of RTCP packets may be obtained, for instance, by adding error protection like Forward Error Correction (FEC) or by employing a connection-oriented protocol for the feedback.

[4]40 byte/packet $*$ 8 bit/byte $*$ 9 packet/frame $*$ 10 frame/second = 28.8 kbit/s

and the corresponding bit rates are found in Appendix B.1, in addition to other configuration parameters.

3.3.3 The number of simulations

For each sequence and parameter set, several simulations were performed according to the method described in Section 3.3.1 in order to ensure results of statistical significance. Video streams that experienced loss in the sequence or picture parameter set or in the first frame were discarded. This was because loss of such information have a major negative effect on the decoded video [3]. Consider the case where parts of a static background is lost in the first frame. The error concealment would in this case be poor since the decoder has no previous frames to use as reference for the temporal/hybrid concealment. FBIR would repair this corruption because the encoder learns that the first frame was damaged. IPLR, however, would not since it never refreshes a static background after the first frame. Hence, to include such simulations would introduce a high variance in the measurements and thus insecurity in the obtained results.

The variances were first used as an indication of statistical significance. 5 "valid" simulations for each parameter set were considered adequate for the Foreman and Soccer sequences. For Conversation 10 "valid" simulations were performed for each parameter set because of higher variances in the measured results. Appendix A shows the resulting 95% confidence intervals.

3.4 Evaluation

The PSNR of the luminance components, denoted Y-PSNR, between the original and decoded video sequences was used as the quality measure. It was chosen because of its simplicity and widespread use. For each test case the average of these Y-PSNR values were calculated to obtain the final objective quality measure. For some simulations the temporal development of the Y-PSNR was also examined.

In addition to Y-PSNR, some information on the performance of the feedback-based ER scheme were extracted from the log-files generated by the encoder and decoder. This information was primarily used to detect implementation errors in the specific components of the scheme, e.g. the error tracking algorithm and the RTP/AVPF timing algorithm. These will not be reproduced in this text, except for results on the temporal properties of RTP/AVPF, which were used to evaluate the performance and potential of the feedback scheme.

The simulation process was executed and supervised manually, while the generation and extraction of evaluation information was carried out automatically. The Y-PSNR was calculated using a small C-program from the EvalVid tool set [39]. For the rest several ad hoc Python-scripts were developed and used.

CHAPTER 4

Results

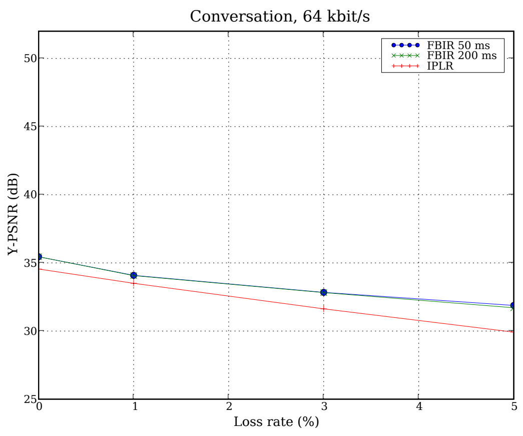
This section provides the results obtained in the simulations. First, the objective measures on video quality are presented. Second, some important visual characteristics and differences for the various experiments are described. Finally, some measures on the transmission behaviour of RTP/AVPF are shown. This section only reproduces the obtained results, while Section 5 provides the analyses.

4.1 Objective comparison

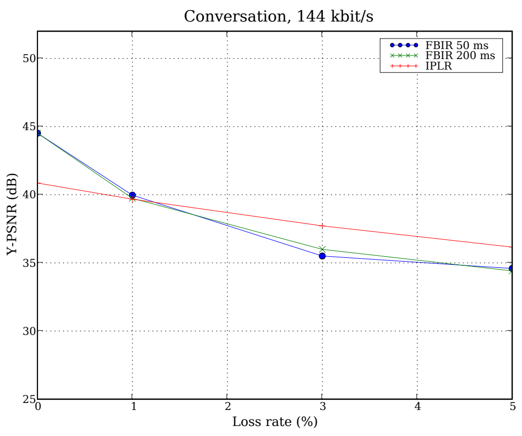
The measured average Y-PSNR values for IPLR and FBIR for 50 ms and 200 ms network latency are compared in Figure 4.1. Figure 4.1(a)- 4.1(c) illustrate their performance for Conversation at 64, 144 and 384 kbit/s. For error free transmission, FBIR is always severely better. At 144 and 384 the difference is approximately 3.5 dB. For 64 kbit/s FBIR is about 1.0 dB higher. At 64 kbit/s FBIR performs better than IPLR for all loss rates. In fact, the difference seems to increase slightly for increased loss rate. Conversely, IPLR outperforms FBIR at 144 and 384 kbit/s when the loss rate exceeds a certain ratio. This threshold is near 1% for 144 kbit/s and 0.5% for 384 kbit/s. From the plots it is evident that, for this sequence, the performance of IPLR is much better than FBIR above this threshold.

Figure 4.1(d) and 4.1(e) shows the measured Y-PSNR values for the Foreman and Soccer sequences at 144 kbit/s, respectively. FBIR yields far better results for these sequences, relative to IPLR, than it does for Conversation at the same bit rate. For Soccer and Foreman FBIR with 50 ms latency is always at least 1 dB higher than IPLR, which is very different from Conversation at 144 kbit/s.

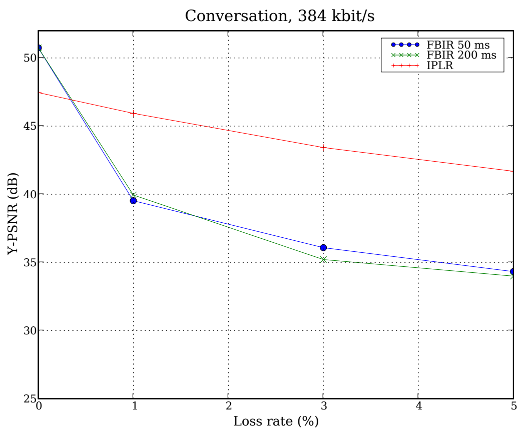
Another important difference between Soccer/Foreman and Conversation, is that the performance of FBIR is clearly reduced when the network latency is



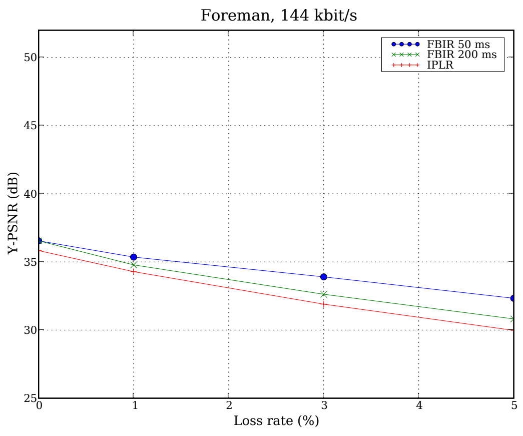
(a) Conversation at 64 kbit/s.



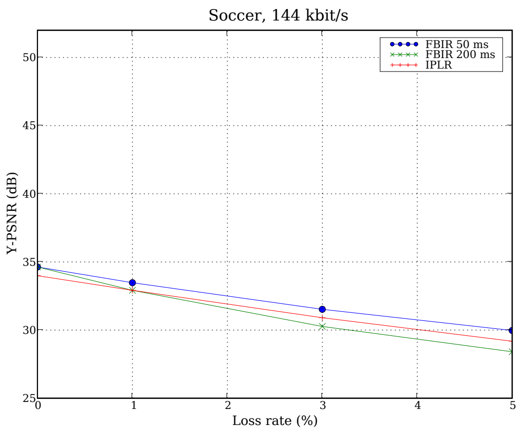
(b) Conversation at 144 kbit/s.



(c) Conversation at 384 kbit/s.



(d) Foreman at 144 kbit/s.



(e) Soccer at 144 kbit/s.

Figure 4.1: Measured average Y-PSNR values with respect to packet loss rate.

increased for both Soccer and Foreman. This is not the case for Conversation. For error free transmissions the obtained Y-PSNR when using FBIR is independent on the network latency, of course. But when a loss rate above 1% is experienced, FBIR with 200 ms latency has a lower Y-PSNR compared with 50 ms latency. At 5% loss the difference is about 1.5 dB, both for Foreman and Soccer. For Soccer, this degradation causes FBIR with 200 ms latency to perform worse than IPLR for loss rates above 1%. For Foreman, on the other hand, FBIR still achieves higher Y-PSNR for all loss rates.

In general, it can be seen that FBIR performs better than IPLR for low bit rates, low loss rates, and for sequences with a medium motion level or more.

It is also interesting to analyze the temporal development of the Y-PSNR for the FBIR and IPLR. Figure 4.2 shows the behaviour when the three bottom GOBs of the 8th frame in the Conversation sequence are lost. FBIR starts out with a significantly higher Y-PSNR until the loss occurs. Then there is a sudden drop in quality for both schemes. Already in the 9th frame IPLR recovers the Y-PSNR to a decent level. From this point it slightly increases until it levels off near the 16th frame. FBIR experience a different behaviour. After the loss, the Y-PSNR continues at its minimum until it suddenly is recovered. With a latency of 50 ms this occurs at the 11th frame, whereas with 200 ms latency the sequence is recovered at frame 14. Notice, however, that the recovered Y-PSNR is somewhat lower than the initial Y-PSNR for both FBIR and IPLR. The reason for this non-perfect recovery is discussed in Section 5.2.3.

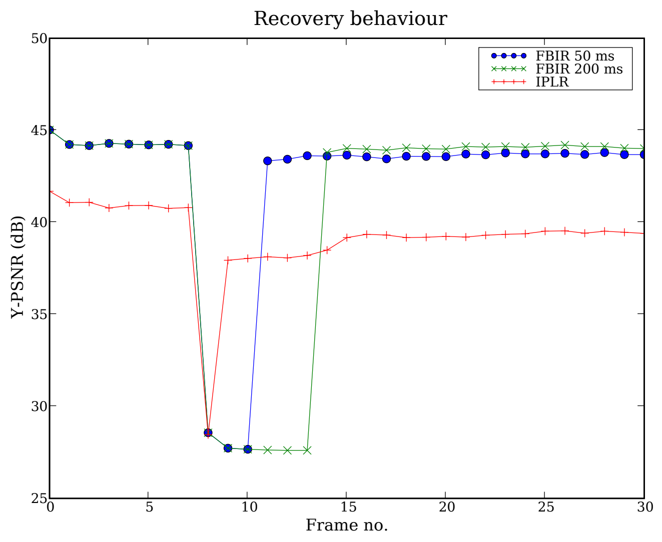


Figure 4.2: Example of the recovery behaviour. The three bottom GOBs of frame 8 is lost. Conversation at 144 kbit/s.

4.2 Visual inspection

Visual inspection confirms many of the objective measures. However, some of the objective results are shown to be misleading with respect to perceptual quality.

For the error free case, sequences encoded with FBIR encounter less distortion than sequences employing IPLR. However, in contrast to the Y-PSNR measurements, the distortion is more visible for low bit rates than for high bit rates. This source coding distortion is seen by loss of small details in low frequency areas. For Conversation at 64 kbit/s there is a clear visual difference between the schemes. Using the IPLR scheme, some of the face features is smeared and disappear, whereas for FBIR these are mostly preserved. There are distortion in other areas as well, but these are not as important for the visual quality as the distortion in the face region. At 144 and 384 kbit/s the source coding distortion in Conversation is not visible for neither FBIR nor IPLR. On the other hand, for Soccer and Foreman at 144 kbit/s there is still some coding distortion for IPLR, though barely noticeable, which is not present when using FBIR.

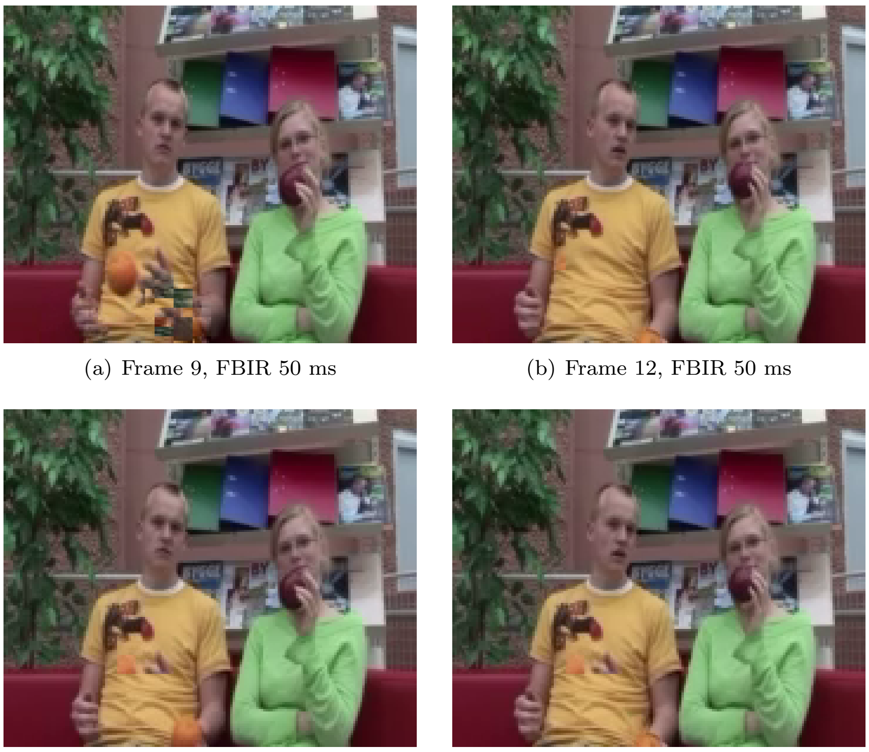
Corruption caused by packet loss is usually only visible in non-static parts of the picture. Errors caused by packet loss is easy to recognize and distinguish from coding distortion. Their appearance have the same characteristics for FBIR and IPLR. Typically, adjacent blocks with clearly unnatural texture suddenly appear. Although they may seem somewhat random at first, the texture of the erroneous blocks have a high correlation with adjacent and previous blocks. The corruption often propagate both temporally and spatially to neighboring regions. This usually coincides with the motion in the video.

There is a distinct difference between FBIR and IPLR regarding the recovery of corrupted areas and thus the error propagation. The FBIR scheme recovers all MBs at the same frame. For 50 ms and 200 ms network latency this normally happens after 2–4 and 5–7 frames, respectively. The exact number of frames is random but also dependent on the bit rate. Larger bit rate decreases the time before recovery, but this is not very noticeable when watching the video in real-time. IPLR refreshes the majority of the corrupted MBs within 1–3 frames. This implies that the FBIR scheme experiences more temporal error propagation, and thus also more spatial propagation. However, some MBs may take a considerably longer time to recover with IPLR. The speed of this process depends on the motion characteristics in the erroneous location before and after the loss occurs. If there is continuously motion in the given area the video is recovered fast. Conversely, if the motion is of a sporadic character, the loss *may* take a longer time to recover from.

The video is not always recovered perfectly. Both FBIR and IPLR sometimes experience that one or more MBs are not refreshed within a reasonable number of frames. Which blocks experience this are different for FBIR and IPLR and is difficult to generalize. Its duration depends primarily on the motions in the video. The effect seems to appear more frequently when IPLR is employed.

Figure 4.3 demonstrates the behaviour described in previous paragraphs. The sequence experience a loss of the three bottom GOBs of frame 8. For FBIR, the corruption is very obvious in frame 9. Using IPLR on the other hand, most of the errors are already repaired one frame after the loss occurred. Only a couple of MBs in the upper part of the corrupted region are still damaged. By accident, this area is not repaired before frame 32. FBIR refreshes the whole corrupted region at frame 12, except for a tiny block which is accidentally recovered after frame 18.

The recovery behaviour is similar for all three test sequences, but the amount of error propagation is different. Conversation encounters less propagation compared with the other two, especially Soccer. In Soccer the errors propagate



(a) Frame 9, FBIR 50 ms (b) Frame 12, FBIR 50 ms

(c) Frame 9, IPLR (d) Frame 12, IPLR

Figure 4.3: Visual artifacts and recovery behaviour. Three bottom GOBs of frame 8 is lost. Conversation, 144 kbit/s.

quickly to a large area and appear more scattered. It is seen that for the FBIR scheme, it is more difficult to recover all corrupted blocks in one instant when the errors propagate like this. However, corrupted regions that fail to intra refresh are recovered accidentally early after because of the motion characteristics of the video.

Increased loss rate naturally has a negative impact on the visual quality for all sequences. It is hard to quantify how bad the effect of increased loss is. However, at 5% loss, Conversation at 64 kbit/s, Foreman and Soccer at 144 kbit/s, it can be seen from visual inspection that the perceived video quality is near unacceptable.

A selection of video examples are provided for the reader in the online archives (see Appendix D).

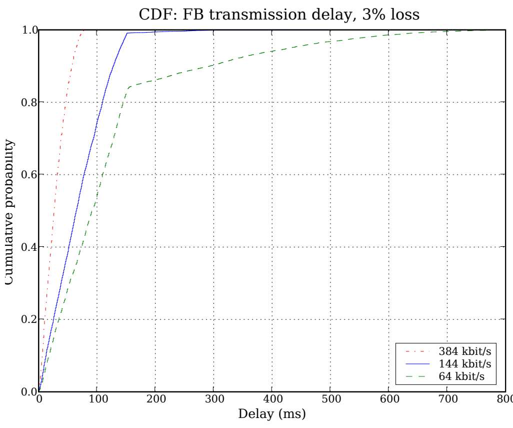
4.3 The temporal properties of RTP/AVPF

The previous results illustrate the FBIR and IPLR schemes as a whole, i.e. the total effect of the algorithms for compression, concealment, error tracking, recovery etc.. This section presents the performance of RTP/AVPF alone with respect to timely feedback.

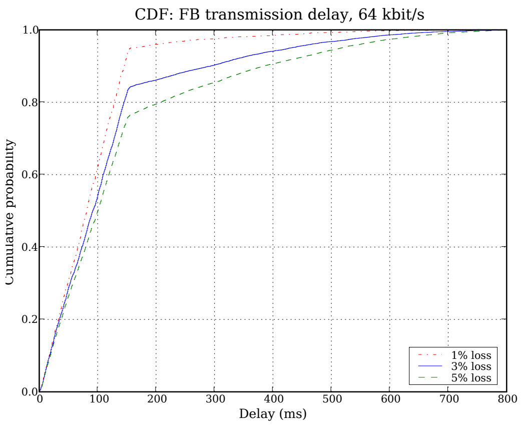
Figure 4.4 shows some Cumulative Distribution Functions (CDFs) for the interval between a loss is detected before the decoder transmits the FB message that reports this loss. No additional delay is included, only the delay generated by the transmission algorithm of RTP/AVPF. The total delay before the encoder receives the FB message consists of this FB transmission delay, two times the network latency ($2 * 50$ ms or $2 * 200$ ms), and the delay until an RTP packet is considered lost (set to 50 ms).

The CDF of FB transmission delays at a 3% loss rate are shown in Figure 4.4(a) for all three session bandwidths. It is evident that as the session bandwidth decreases, the FB transmission delay is statistically increasing. At 384 kbit/s, all FB messages are sent before the defined Early FB maximum delay of 150 ms (see Section 3.2). In fact, all loss reports are sent within approximately 85 ms after the loss detection. At 144 kbit/s, approximately 98% of the FB messages are transmitted before 150 ms. At 64 kbit/s, less than 85% are transmitted before 150 ms. Some FB messages are not sent until 7-800 ms after the loss detection.

Figure 4.4(b) illustrates how the FB transmission delay depends on the packet loss rate at 64 kbit/s. It is clear that the average delay is larger for higher loss rates. For 1, 3 and 5% loss, there is approximately 95%, 84% and 77% of the FB messages, respectively, that are transmitted before 150 ms after the loss detection. However, the maximum delay does not increase noticeably. By comparing this result to equivalent measures for higher bit rates (not illustrated), it is seen that the differences in FB transmission delays are less. At 384 kbit/s the effect of increased loss rate is negligible for loss rates up to 5%.



(a) Various bit rate, 3% packet loss



(b) 64 kbit/s, various packet loss

Figure 4.4: CDF for the FB transmission delay, i.e. the interval between the decoder's detection of a loss and its transmission of the FB message.

CHAPTER 5

Discussion

This section evaluates the results obtained in Section 4. First, RTP/AVPF and its temporal properties and potential as part of feedback-based ER schemes are analyzed. Then, the performance of FBIR and IPLR is compared and discussed. Finally, sources of error and how they may have affected the results are considered. This also includes suggestions for improvement of the FBIR scheme and implementation.

5.1 On the performance of RTP/AVPF

In this section the performance of RTP/AVPF as the feedback protocol in FBIR is evaluated. A general discussion on the usage of this protocol in feedback-based error control is also carried out.

5.1.1 The FB transmission interval

RTP/AVPF, as described in Section 2.3.2, is responsible for sending timely feedback with helpful information to the encoder. From the results in Section 4.3 it is seen that the transmission of FB messages is affected by the RTP session bandwidth and the packet loss rates. There are also other variables that influence the feedback transmission intervals, e.g. the number of senders/receivers. Some of them will be discussed later in this section.

From Figure 4.4(a) it is evident that the average transmission delay decreases when the RTP session bandwidth is increasing. This is expected since the feedback bandwidth is set to a fixed ratio of the session bandwidth. Hence, when the session bandwidth increases the transmission rules allow shorter intervals between each FB message (or regular RTCP packet). A more frequent feedback is of course advantageous since it enables errors to be repaired faster.

The above observation can also be seen by inspecting the timing interval algorithm [4] [5]. In practice, this interval affects the maximum number of RTCP packets (early or regular) that are allowed to be transmitted over a period of time. On the average, only one packet may be sent each interval. The calculated interval is proportional according to the formula

$$t \propto \frac{\text{avg\_rtcp\_size} * \text{members}}{\text{rtp\_bw} * \text{rtcp\_frac}}, \tag{5.1}$$

where *avg_rtcp_size* is the moving average size of the last 16 RTCP packets, *members* is the sum of senders and receivers in the current RTP session (or only the number of receivers if $receivers > 4 * senders$), *rtp_bw* is the RTP session bandwidth, and *rtcp_frac* is the fraction of the session bandwidth used for RTCP. The standard recommends that *rtcp_frac* is fixed at 5%. If this is to be followed, *rtcp_frac* may be considered a constant. *avg_rtcp_size* depends on which RTCP packet types are included in the RTCP packets and their contents. This is application specific and should be kept at a minimum to gain short timing intervals. When the number of senders and/or receivers increase, it means that more endpoints must share the available RTCP bandwidth. Hence, an increase of *members* leads to longer timing intervals. However, only simulations assuming one sender and one receiver was performed in this study. At last, as already stated, increased RTP session bandwidth *rtp_bw* yields shorter timing intervals and enables more frequent feedback.

Figure 4.4(b) shows that the average FB transmission delay also increase when the loss rate increases, even though the packet loss rate does not have any direct impact on the timing algorithm. When more losses occur, the number of needed FB messages grows. Since the timing interval remains unchanged and maximum one RTCP packet is sent each interval on the average, a lower ratio of the losses is allowed to be reported with Early FB messages. Those which are not reported with Early FB messages must wait until the next regular RTCP packet. Hence, the average transmission delay increases. Figure 4.4(b) clearly shows this effect at 64 kbit/s. However, it has been stated earlier that the effect is negligible at 384 kbit/s. This is because at high rates the FB transmission interval is short enough to report most losses almost immediately. Either as Early FB message or as part of a regular RTCP packet that is transmitted before the maximum Early FB transmission delay (which was set to 150 ms in this implementation).

5.1.2 The feedback information

The NACK FB message applied in the FBIR scheme is very general as it is independent of the RTP payload. It is easy to implement, but limits the possibilities within the ER scheme. However, RTP/AVPF also provides other sorts of feedback information which may be useful for error control.

When using NACK, it is necessary to map the received NACKs to MB and frame numbers. This is only possible by having a fixed number of MBs in each RTP packet ("slice mode"). As described in Section 2.5, this is employed in the FBIR scheme. However, this degrades the coding efficiency and may not be desired. RTP/AVPF provides some payload-specific FB message types that solve this matter, such as Picture Loss Indication, Slice Loss. Since both FBIR and IPLR applied "slice mode" in the simulations, their relative performance should not be affected by the reduced coding efficiency. Hence, the performed comparisons is expected to be valid for similar FB message which do not enforce "slice mode".

As described in Section 2.3 there are other types of feedback information that can be provided, either by RTP/AVPF alone or in combination other standards such as AVPF/CCM and H.271. These will not be repeated here. Together these standards enable information that should be sufficient for all feedback-based ER schemes. If the "native" message types do not provide the necessary information, RTP/AVPF's application-specific FB messages can always be employed to send custom feedback information.

5.2 Evaluation of FBIR and IPLR

The number of implementation-specific parameters, testing parameter sets, and different test sequences make comparison of FBIR and IPLR a complex task. For a clearer comparison the evaluation is broken down in sections. Each section examines the effect of one or a few parameters while keeping the others fixed. The discussion is based on the objective measures as well visual observations.

5.2.1 Coding efficiency and error free transmission

The FBIR scheme always achieve higher Y-PSNR than IPLR in error free environments, as seen from Figure 4.1. This is expected since IPLR adds redundant error protection, while FBIR does not enforce extra intra coded MBs when no errors are reported. Naturally, network latency does not matter in this case.

The severe difference in Y-PSNR between FBIR and IPLR does not always correspond to the difference in perceptual quality. The difference is more visible at low bit rates[1]. This is because PSNR does not consider characteristics of the HVS. Apparently, the type of distortion generated at high bit rates and not visible to the human eye, is still detected by PSNR and causes the objective measurements to differ from perceptual quality. This occurs for Conversation at 144 and 384 kbit/s. Based on this observation, one can say that the redundancy added by IPLR does not reduce the perceptual quality as long as the bit rate is sufficiently high so that the coding distortion described in Section 4.2 does not appear. However, if this is not the case, FBIR performs noticeably better for error free transmissions.

5.2.2 Visual artifacts

The artifacts generated by packet losses were described in Section 4.2. In order to explain the appearance of this distortion, it is necessary to look at both the encoding and decoding scheme. First, MBs are the basic building units of the coding process. Thus, an error affect at least a whole MB. Second, each RTP packet is forced to contain exactly one GOB. Hence, at least one whole "row" of MBs is lost for each packet loss. In the case of bursts, consecutive GOBs are corrupted. This is why the corruption appear as adjacent block shaped regions.

[1]"Low bit rate" is, of course, a relative concept which depends on video format and characteristics.

Third, errors usually appear in motion active regions because JM's hybrid concealment scheme normally conceals lost data on static areas perfectly. Finally, the spatial and temporal error propagation are caused by inter prediction. Obviously, when the decoder predict a block based on erroneous values the predicted block will also become corrupted.

5.2.3 Error recovery behaviour

Before evaluating the total performance of FBIR and IPLR in error prone environments, the recovery behaviour of the two schemes is presented. The awareness of this temporal behaviour is important to understand the differences in performance discussed later.

As evident from Figure 4.2, the FBIR scheme preserves a higher quality while the video is error free, because of the advantageous coding efficiency described previously. When an error occur, here in frame 8, FBIR and IPLR experience a sudden drop in quality. Since FBIR and IPLR loose the same information and use the same concealment strategy, the damage is almost the same for the two schemes. This corruption is more prominent than the coding distortion. Hence, FBIR and IPLR encounter approximately the same quality in the frame where the loss occurs.

As seen from both the Y-PSNR measurements and the visual inspection, the corruption propagates over several frames when using FBIR. This is because the encoder assumes error free transmission, i.e. no MBs are intra refreshed, until it receives the feedback. Naturally, longer network latency results in longer delay before the loss is reported to the encoder. This postpones the video recovery, which is seen in Figure 4.2 by the difference in time of recovery for FBIR with 50 and 200 ms latency. The average response in this case increases with $(2 * (200 \text{ ms} - 50 \text{ ms})) = 300 \text{ ms} = 3$ frames. When the encoder becomes aware of the error, it refreshes the whole corrupted area at once. This instantly removes all corruption if the error tracking algorithm and the concealment of static areas works perfectly.

IPLR, on the other hand, performs the same procedure regardless of network latency, since it does not employ feedback. Contrary to FBIR, the recovery process often lasts a few frames, as pointed out in Section 4.1 and 4.2. This is because of IPLR's motion-based intra refresh algorithm. Since (usually) not all MB locations contain motion all the time, IPLR will intra refresh MBs on different intervals. Hence, all MBs are not intra refreshed in the same frame. Instead, the corrupted region encounters a gradual refresh.

The visual inspection revealed one problem with both FBIR and IPLR: Some corrupted MBs are not recovered within the expected time. For FBIR this occurs primarily when errors are not properly tracked. There are two reasons for this. First, precision was reduced in favor of speed by using the fast error tracking algorithm. Second, and probably more important, the implementation used predicted MVs instead of the real MVs of the current frame since these were not available. After the intra refresh, the encoder assumes that all errors are corrected. Hence, untracked errors will persist until the corrupted MBs are accidentally intra coded according to the normal encoding scheme, or until a new error occurs at the same location and gets repaired.

In the case of IPLR, the reason for uncorrected errors is unlucky timing of motion characteristics, "skipped" MBs and the occurrence of the errors. If an erroneous MB becomes visible and is/was encoded in skip mode in the next/previous frame, the MB is probably not refreshed until many frames later since the IPLR counter is reset to "-10" by the skip mode.

Examples of imperfect recovery as discussed here is seen in Figure 4.3(b) and 4.3(d). This is also the reason why the Y-PSNR in Figure 4.2 is not as high after recovery as it was before the error. By accident, FBIR seems to experience less of this negative effect in when the network latency is 200 ms for this particular example.

5.2.4 The effect of increased bit rate

Obviously, increased bit rate implies improved quality. But the relative performance between schemes may change, in regards to both coding efficiency and error robustness. From Figure 4.1(a)-4.1(c) it is evident that IPLR usually improves its performance relative to FBIR when the bit rate increases and there is loss.

For Conversation, 64 kbit/s is considered a medium to low bit rate. For this rate and sequence FBIR performs better than IPLR. This is primarily because FBIR encounters less source coding distortion. At high bit rates the source coding distortion becomes very low, both for IPLR and FBIR. The corruption caused by lost packets thus becomes (more) dominant to the source coding distortion. Hence, it is more important to have high robustness, that is, to recover rapidly from errors. IPLR is in general more robust than FBIR since it on average intra refreshes corrupted MBs earlier than FBIR, as discussed in Section 5.2.3. This is true even if higher bit rates implies shorter feedback delays for FBIR. Because of this, IPLR performs better than FBIR at high bit rates when the loss rate is greater than a certain value.

It is also interesting to note that the objective quality for Conversation when using FBIR at 144 and 384 kbit/s, is almost equal for loss rates at 1% or higher. This means that the reduced source coding distortion gained by increasing the bit rate, may be insignificant for high rates when the loss exceeds a certain ratio. This observation is another argument for stating that the corruption caused by lost packets are dominant at high rates. It also indicates that the increased feedback bandwidth, which results in feedback with less delay, does not improve the performance of RTP/AVPF significantly in this low motion case. It may, though, for other rates and sequences.

5.2.5 The effect of increased network latency

As already explained, if feedback is employed, larger network latency implies higher average delay before the encoder receives the feedback information. This obviously does not affect IPLR. For FBIR, however, this will in most cases degrade the quality. It will take longer time before the corruption is repaired. Consequently, the errors may also spread to a spatially larger area. An effect of this is the difference in Y-PSNR between FBIR with 50 and 200 ms latency for the Foreman and Soccer sequences (Figure 4.1(d) and 4.1(e)). However, this reduced performance is not always notable. For Conversation (Figure 4.1(a)-4.1(c)) FBIR obtains approximately the same Y-PSNR for both latencies. The reason for this is the difference in motion characteristics of the sequences. Foreman and Soccer has medium to high motion activity, which results in poor error concealment and severe error propagation. Conversation, on the other hand, has a limited degree of motion and a large portion of the picture is static. Hence, errors are often concealed well and propagate less. As a result, the extended feedback delay does not lead to a significant quality reduction.

Surprisingly, for some parameter sets on the Conversation sequence, FBIR achieves slightly better Y-PSNR with 200 ms latency. This is probably an effect of the random loss patterns and the low motion activity in this sequence. This source of error is discussed in Section 5.3.

5.2.6 The effect of increased loss rate

Naturally, increased loss rate degrades quality. It is more interesting to examine the relative differences between FBIR and IPLR with respect to increased packet loss.

For IPLR the measurements show that the reduction of Y-PSNR with respect to increased loss rate is near constant. That is, the Y-PSNR encounters the same degradation if the loss rate is increased from 0 to 1% or from 4 to 5%. This is because the IPLR algorithm is independent of loss patterns and other network statistics. It performs intra refresh only based on the motion characteristics of the sequence. Hence, it roughly gains the same error robustness for all simulated loss rates.

In contrast to IPLR, FBIR does *not* encounter a constant decrease in Y-PSNR for increased loss rate, especially not for high bit rates. There is, for instance, a higher drop in objective quality from 0 to 1% loss compared with from 4 to 5% loss. This can be explained by considering the following very simplified scenario: At low loss rates, most errors are allowed to be reported by Early FB messages. Thus, most errors are corrected after approximately the same number of frames and thus generate about the same amount C of corruption. A small increase X in the number of losses will generate $X * C$ more corruption. When the loss rate is high, the situation is different. Due to the calculated transmission interval, the ratio of losses reported by Early FB becomes lower, as described in Section 5.1.1. This implies that one FB message must report several losses, on the average. Thus, each intra refresh repairs corruption caused by several losses. Logically, each loss will then contribute to *less* than the amount C of corruption, on the average. A small increase X in the number of losses will thus reduce the quality less compared with the low loss rate case above. Hence, changes in the loss rate affect FBIR less when the loss rate is high.

5.2.7 The effect of motion characteristics

The fact that the relative performance of FBIR and IPLR is different for the three test sequences indicates that their motion characteristics are important for the obtained quality.

In sequences with high motion activity the errors are harder to conceal and the propagation is more severe. This applies for both FBIR and IPLR. The main reason FBIR performs better than IPLR for some sequences and vice versa, is a change in coding efficiency. Since IPLR is proactive and entirely motion-based, it is more affected than FBIR when the test sequence is changed. For sequences with a higher motion level, more MBs are frequently intra coded. Thus, IPLR "wastes" even more bits on protection of information that may not be lost. FBIR still only intra refreshes regions that actually are corrupted. Hence, it is expected that FBIR performs better than IPLR for sequences with a motion activity above medium. This is confirmed by comparing the results for Foreman, Soccer and Conversation at 144 kbit/s in Figure 4.1. If Foreman and Soccer were encoded on a higher rate, say 384 kbit/s, it is expected that the results would be more similar to those for Conversation at 144 kbit/s. This is because sequences with high motion activity, e.g. Foreman and Soccer, needs higher bit rate to encounter the "high rate" effects described in Section 5.2.4.

To sum up the evaluation of FBIR and IPLR: Which one is better of the two ER schemes depends on many factors, the most important being the motion characteristics of the sequence, applied bit rate and the packet loss rate. FBIR is a more general ER scheme than IPLR and is suited for many applications. It always outperforms IPLR when the loss rate is very low. FBIR also yields better results for sequences with a medium motion activity or higher, or if the bit rate is low. IPLR, on the other hand, is better suited for sequences with low motion activity, i.e. a large portion of the picture is static, as long as the bit rate is high enough for IPLR to encode the video without severe coding distortion. The latter is a typical setting for video communication systems, which IPLR was designed for.

5.3 Sources of error

There are some insecurity in the generated results and observations. This section explains the elements that contribute most to this.

As already stated in Section 2.8, PSNR does not coincide well with HVS characteristics, and thus not always with perceptual quality; the HVS does not evaluate a picture pixel by pixel. For a given video sequence, high PSNR indicates good quality and vice versa, but the PSNR can not be used as an "absolute" measure of perceptual quality. On the other, the FBIR and IPLR scheme produces very similar corruption artifacts. Therefore the PSNR measure should at least give a good indication on which scheme is visually better.

The randomness in the loss patterns generated by the network emulator may also cause misleading results. Such stochastic processes require many simulations to generate results of statistical significance. Appendix A provides a table of the 95% confidence interval for the measured Y-PSNR values. The Soccer and Foreman simulations were shown to generate a very narrow interval, especially compared to the simulations on the Conversation sequence. Thus, the results obtained for these two sequences may be considered very reliable. For Conversation, on the other hand, some confidence intervals are wider than desirable, i.e. there are more insecurity in the results. A result that probably is erroneous as a consequence of this effect were mentioned in Section 5.2.5: FBIR with 200 ms latency performed better than with 50 ms latency for the Conversation sequence at 144 kbit/s and 3% loss rate. This error is caused by the combination of Conversation's motion characteristics and the spatial and temporal distribution of lost data. A simulation that encounters most of its losses in the static area of the picture will gain higher Y-PSNR compared with a simulation that experience most losses in regions with high motion activity, despite the longer network latency. More simulations for each parameter set would reduce the risk for such errors. Soccer and Foreman are not severely affected since their motion activity is spread out over the entire picture. Hence, the spatial position of the lost data is not equally important for these two sequences.

Some implementation-specific approximations of the error tracking algorithm and the detection of static MBs were necessary. These were described in Section 3.2. The use of predicted MVs instead of the real MVs of the current frame clearly may introduce errors. This can cause some corrupted MBs to not be recovered, even though they should according to the FBIR scheme. The effect is that the Y-PSNR will be reduced over time. Some reduction of the Y-PSNR is expected during long sequences since we do not employ precise error tracking, but the approximated implementation amplifies this effect. These are not errors in the results, per se, but degrades the performance of the FBIR scheme, while IPLR is unaffected. An example of this effect is illustrated in Figure 5.1. It is a plot of the moving average (applying a window of 50 frames on all 10 simulations) of Y-PSNR for Conversation at 144 kbit/s and 1% loss. It is obvious that FBIR suffers from a significant reduction in performance over time. However, shorter sequences would reduce this effect. But this would again amplify the uncertainty introduced by using a random loss generator. It is difficult to say how much of this degradation is caused by the implementation-specific approximations as opposed to the intentional simplifications in the scheme. Without doubt, an accurate implementation would improve the results of the FBIR scheme.

Figure 5.1 also shows the corresponding performance of an alternative FBIR scheme, denoted FBIR-2. The difference from the regular FBIR scheme is that the encoder insert an IDR picture every 300th frame. Thus, the video is completely resynchronized every 30th second, which severely reduces the effect of the suboptimal error tracking. The bit rate increases with about 2.5 kbit/s on average, which is acceptable compared to the gained quality. For Conversation at 144 kbit/s with 1% loss the gain in Y-PSNR by using FBIR-2 instead of FBIR is more than 2 dB. Unfortunately, there was not enough time to perform more simulations with FBIR-2. However, this result confirms that an accurate implementation of the FBIR scheme would perform significantly better.

5.4 Other considerations

The simulations employed sequences of QCIF resolution and 10 frames per second. Most video communication systems, however, at least on wire-line, transmits video on CIF resolution or higher and may also use more frames per second. Consequently, the bit rate is normally increased, more RTP packets are sent and the packet size is larger. This should not have significant impact on IPLR. The FBIR scheme could, however, experience some change in performance. This statement is based on the fact more FB messages will be needed since the number and frequency of losses increase (assuming that the packet loss rate is constant). On the other hand, the available feedback bandwidth will also increase, which enables more frequent transmission of FB messages. It is difficult to say the exact effect of this without a more thorough examination. Unfortunately, the resources available for this study did not enable such experiments.

Another factor to consider is that most real-life system utilize rate control to prevent network congestion. An encoder that suddenly intra codes most MBs of a frame will encounter a severe increase in the bit rate for a short period

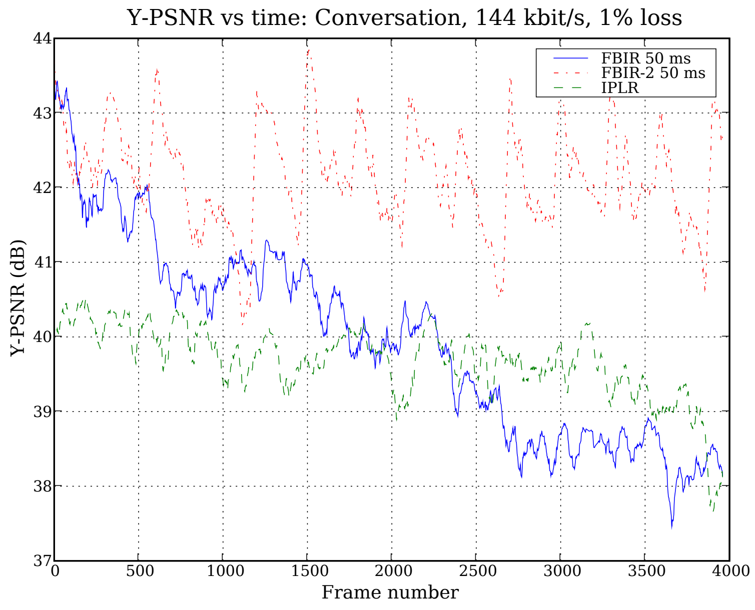


Figure 5.1: Development of Y-PSNR over time. FBIR-2 resynchronizes the video every 300th frame, which significantly reduces the degradation encountered by FBIR.

of time. This may lead to longer queuing delay and more packet losses in the network. The rate control aims to minimize such rate bursts. Traditionally, this is carried out by adapting encoding parameters such as quantization, frame rate etc.. However, this approach may reduce the visual quality. For the FBIR scheme an alternative may be to distribute the intra refresh over several frames at the cost of a somewhat longer recovery interval. The FBIR scheme is more prone to rate controlling actions than IPLR because of its sudden recovery of the whole corrupted area.

CHAPTER 6

Future work

As explained in Section 5, the FBIR scheme suffered from an implementation-specific approximation. It would be interesting to test how well FBIR is able to perform with an implementation exactly like intended. That is, to alter the encoding scheme to enable the use of only real MVs in the error tracking and static MB detection. A significant increase in Y-PSNR is expected. Alternatively, a gain in performance may also be achieved by periodically resynchronizing the video, as shown in Section 5.3. Simulations should be performed to measure more thoroughly the effect of such resynchronizing, and to find an optimal resynchronization interval.

This study did not experiment by tweaking the implementation-specific parameters in the RTP/AVPF standard. It is desirable to find more optimal parameters in order to increase the performance of feedback-based ER schemes utilizing RTP/AVPF. Optimal parameters will depend on the application, but research should be carried out to examine the effect of changing different parameters for various applications.

Simulations utilizing sequences of higher resolution and more frames per second are also desirable. This is to test how RTP/AVPF responds to increased traffic and thus more frequent losses, and its effect on the ER scheme(s). In order to enable such simulations it is necessary to either use a more powerful processor for the encoder; use another encoder in place of JM that is faster; or to use another testing framework which allows offline encoding while still generating realistic feedback as if online simulation is performed.

Intra refresh is only one approach to utilize the available feedback information using RTP/AVPF. As described, RTP/AVPF and its related protocols, e.g. H.271, provides feedback information that may enable many novel feedback-based ER schemes. In addition, the potential of already known ER schemes, such as RPS, in combination with RTP/AVPF should be examined.

CHAPTER 7

Conclusion

In this study the new (2006) feedback protocol RTP/AVPF and its use in ER schemes for real-time low-delay video systems have been evaluated. A feedback-based intra refresh scheme, FBIR, utilizing RTP/AVPF was developed and compared with IPLR. IPLR is a motion-based intra refresh scheme which does not employ any feedback information. The two ER schemes were compared for various network environments, bit rates and video sequences.

RTP/AVPF is based on RTCP as the underlying feedback protocol, and may thus be applied to all applications using RTP. It is fairly easy to implement since it only modifies the RTCP timing algorithm and adds new RTCP message types. RTP/AVPF may be used in combination with other standards, such as H.271, to extend the available feedback information. The results prove that RTP/AVPF is able to provide feedback early enough to be useful in error control for many rates and network conditions. Hence, RTP/AVPF enables timely feedback for use in a wide range of multimedia applications.

The comparison of the two ER schemes shows that FBIR always performs better than IPLR for error free transmissions. FBIR achieves higher quality in other situations as well, such as for very low loss rates, low or medium bit rates, and for sequences with high or medium motion activity. Conversely, IPLR is better suited for video sequences containing less motion, encoded at high bit rates when the loss rate exceeds a certain threshold, typically about 1%. The results are a consequence of that FBIR gains a medium robustness and high coding efficiency, in contrast to IPLR's high robustness and low coding efficiency. These properties are caused by differences in the intra refresh strategies. Basically, IPLR inserts a high number of intra coded MBs to improve the robustness in case losses will occur. FBIR, on the other hand, reacts only to reported errors and performs intra refresh only to correct these. Hence, IPLR usually repair the erroneous regions faster, while FBIR introduce less redundancy and thus gains better coding efficiency. FBIR's performance may, however, be reduced by factors such as increased network latency and the number of receivers, because these factors increase the feedback delay of RTP/AVPF.

In error free environments, FBIR does not decrease the coding efficiency significantly compared with a non-robust encoding scheme. Thus, FBIR is a good compromise between coding efficiency and error robustness. All real-time video systems that benefit from immediate feedback should therefore strongly consider to employ FBIR or similar feedback-based ER schemes.

Bibliography

[1] ITU-T Rec. H.264: Advanced video coding for generic audiovisual services; also ISO/IEC 14496-10:2005, Coding of audio-visual objects Part 10: Advanced Video Coding, 2005.

[2] TANDBERG. *TANDBERG and packet loss*, D50165, Rev 2.0 edition.

[3] Stian Selnes. Evalution of IPLR and other error resilience schemes. An in-depth report as part of the study for a master's degree, December 2006.

[4] J. Ott, S. Wenger, N. Sato, C. Burmeister, and J. Rey. Extended RTP Profile for Real-time Transport Control Protocol (RTCP)-Based Feedback (RTP/AVPF). RFC 4585 (Proposed Standard), July 2006.

[5] H. Schulzrinne, S. Casner, R. Frederick, and V. Jacobson. RTP: A Transport Protocol for Real-Time Applications. RFC 3550 (Standard), July 2003.

[6] Pao-Chi Chang and Tien-Hsu Lee. Precise and Fast Error Tracking for Error-Resilient Transmission of H.263 Video. *IEEE Transactions on Circuits and Systems for Video Technology*, 10(4):600–656, June 2000.

[7] Cristina Gomila and Peng Yin. New features and applications of the H.264 video coding standard. In *International Conference on Information Technology: Research and Education, 2003. Proceedings. ITRE2003.*, pages 6 – 10. IEEE, August 2003.

[8] Thomas Wiegand, Gary J. Sullivan, Gisle Bjøntegaard, and Ajay Luthra. Overview of the H.264/AVC Video Coding Standard. *IEEE Trans. Circuits Syst. Video Techn.*, 13(7):560 – 576, 2003.

[9] J. Ostermann, J. Bormans, P. List, D. Marpe, M. Narroschke, F. Pereira, T. Stockhammer, and T. Wedi. Video coding with H.264/AVC: tools, performance, and complexity. *Circuits and Systems Magazine, IEEE*, 4(1):7 – 28, 2004.

[10] Stephan Wenger. H.264/AVC over IP. *Circuits and Systems for Video Technology, IEEE Transactions on*, 13(7):645–656, 2003.

[11] D. Ferrari. Client requirements for real-time communication services. RFC 1193 (Informational), November 1990.

[12] Till Halbach and Steffen Olsen. Error robustness evaluation of H.264/MPEG-4 AVC. In *Visual Communications and Image Processing 2004. Proceedings of the SPIE, Volume 5308, pp. 617-627 (2004)*, pages 617–627. SPIE, Janauary 2004.

[13] Till Halbach. The H.264 Video Compression Standard. In *Proceedings of Nordic Signal Processing Symposium (NORSIG)*. NORSIG, October 2003.

[14] Sunil Kumar, Liyang Xu, Mrinal K. Mandal, and Sethuraman Panchanathan. Error Resiliency Schemes in H.264/AVC Standard. *Journal of Visual Communication and Image Representation*, 17(2), April 2006.

[15] Franco Chiaraluce, Lorenzo Ciccarelli, Ennio Gambi, and Susanna Spinsante. Performance Evaluation of Error Concealment Techniques in H.264 Video Coding. In *Picture Coding Symposium 2004*. University of California Davis, December 2004.

[16] B. Girod and N. Farber. Feedback-based error control for mobile video transmission. *Proc. IEEE*, 87(10):1707–1723, October 1999.

[17] S. Wenger, U. Chandra, M. Westerlund, and B. Burman. Codec Control Messages in the RTP Audio-Visual Profile with Feedback (AVPF). Internet Draft, `http://www.ietf.org/internet-drafts/draft-ietf-avt-avpf-ccm-04.txt`, March 2007.

[18] ITU-T Rec. H.271: Video back channel messages for conveyance of status information and requests from a video receiver to a video sender, 2006.

[19] ITU-T Recommandation H.245: Control protocol for multimedia communication, 2006.

[20] ITU-T Recommandation H.323: Packet-based multimedia communications systems, 2006.

[21] K. El Maghraoui and T. Rachidi. Towards building h.323-aware 3g wireless systems: H.323 control loops and applications adaptation to wireless link conditions. In *Proceedings of the 5th World Multiconference on Systemics, Cybernetics and Informatics (SCI 2001)*, volume 16, pages 106–113, July 2001.

[22] Y. Wang, S. Wenger, J. Wen, and A. Katsaggelos. Review of Error Resilient Coding Techniques – Real-Time Video Communications over Unreliable Networks. *IEEE Signal Processing Magazine*, 17(4), July 2000.

[23] Li Zhu, Hao Chen, and Xinyu Yang. An error control mechanism based on adaptive intra-frame refreshment. In *Second International Conference on Image and Graphics,Proceedings of the SPIE Volume 4875 (2002)*. SPIE, 2002.

[24] Ye-Kui Wang, Chunbo Zhu, and Houqiang Li. Error resilient video coding using flexible reference frames. In *Visual Communications and Image Processing 2005. Proceedings of the SPIE, Volume 5960, pp. 691-702 (2005)*, pages 691–702. SPIE, Jul 2005.

[25] Hong-Bin Yu, Ci Wang, and Songyu Yu. A novel error recovery scheme for H.264 video and its application in conversational services. *IEEE Transactions on Consumer Electronics*, 50(1):329–334, Feb 2004.

[26] H.264/AVC reference software JM 11.0. `http://iphome.hhi.de/suehring/tml/download`.

[27] Keng-Pang Lim, Gary Sullivan, and Thomas Wiegand. *Text Description of Joint Model Reference Encoding Methods and Decoding Concealment Methods, JVT-N046*. Joint Video Team (JVT) of ISO/IEC MPEG and ITU-T VCEG, January 2005.

[28] Alexis Michael Tourapis, Karsten Sühring, and Gary Sullivan. *Revised H.264/MPEG-4 AVC Reference Software Manual, JVT-Q042*. Joint Video Team (JVT) of ISO/IEC MPEG and ITU-T VCEG, Rev. 5 edition, October 2005.

[29] Z. Wang, H. R. Sheikh, and A. C. Bovik. *The Handbook of Video Databases: Design and Applications*, chapter “Objective video quality assessment”, pages 1041–1078. CRC Press, September 2003.

[30] Q2S – Centre for Quantifiable Quality of Service in Communication Systems, NTNU. `http://www.q2s.ntnu.no`.

[31] Odd Inge Hillestad, Bjørnar Libak, and Andrew Perkis. Performance Evaluation of Multimedia Services Over IP Networks. In *Proceedings of the IEEE International Conference on Multimedia and Expo (ICME)*, pages 1464 – 1467, Amsterdam, The Netherlands, July 2005.

[32] Empirix PacketSphere Network Emulator. `http://www.empirix.com`.

[33] Endace. `http://www.endace.com`.

[34] AviSynth. `http://www.avisynth.org`.

[35] VirtualDub. `http://www.virtualdub.org`.

[36] Stephan Wenger. Common Conditions for wire-line, low delay IP/UDP/RTP packet loss resilient testing, September 2001. ITU-T VCEG Document VCEG-N79r1.

[37] Stephan Wenger. Proposed Error Patterns for Internet Experiments, including Appendix 11, October 1999. ITU-T VCEG Document Q15-I-16.

[38] Athina Markopoulou, Fouad Tobagi, and Mansour Karam. Loss and Delay Measurements of Internet Backbones. *Computer communications*, 29(10):1590–1604, 2006.

[39] EvalVid 1.2. `http://www.tkn.tu-berlin.de/research/evalvid/`.

[40] Ronald E. Walpole, Raymond H. Myers, Sharon L. Myers, and Keying Ye. *Probability & Statistics for Engineers & Scientists*. Prentice Hall, 2002.

[41] Wireshark. `http://www.wireshark.org/`.

[42] Apple QuickTime. `http://www.apple.com/quicktime`.

[43] Stian Selnes. Online archive for "Feedback-based Error Control for H.264. `http://www.pvv.ntnu.no/~stianse/master`. Username: master, password: thesis.

APPENDIX A

Result details

This appendix provides more details on the main results presented in Section 4. The Y-PSNR numbers for each parameter set and test sequence are listed. These were used to plot the performance of the ER schemes with respect to the encountered loss rate, in Figure 4.1. The statistical significance of these numbers are illustrated by calculating the 95% confidence interval.

A.1 Confidence intervals

Table A.1 lists the calculated averages of the measured Y-PSNR values, which are plotted in Figure 4.1. It also shows the corresponding 95% confidence intervals.

A confidence interval quantifies the uncertainty of estimates. The 95% interval gives the range where one can be 95% certain that the true mean value lies within. Analyzing the numbers in Table A.1 it is obvious that there is more uncertainty related to some simulations compared to others. Especially for the Conversation sequence the confidence interval is sometimes larger than desirable. However, the simulations with highest uncertainty also have a large difference between the Y-PSNR estimates of FBIR and IPLR. Hence, there should be clear which of the two schemes are better for these parameters, even if the estimated Y-PSNR is somewhat unreliable.

The confidence intervals were calculated by assuming the Y-PSNR values to have a normal distribution. However, since only 5 or 10 simulations were performed for each parameter set, the Student's t-distribution was employed in the calculations. The confidence interval was calculated according to [40]:

$$P(Y - t_{\alpha/2,n-1}\frac{S}{\sqrt{n}} < \mu < Y + t_{\alpha/2,n-1}\frac{S}{\sqrt{n}}) = 1 - \alpha, \qquad \text{(A.1)}$$

where Y is the average Y-PSNR taken over n video sequences, S is the estimated standard deviation, α corresponds to the confidence of $1-\alpha$ (e.g. $1-0.05 = 0.95$), and $t_{\alpha/2,n-1}$ is a constant for the Student's t distribution found by table look-up.

				FBIR		IPLR	
Sequence	Target rate	Latency	PLR	Y-PSNR	$CI_{95\%}$	Y-PSNR	$CI_{95\%}$
Conversation	64 kbit/s	50 ms	0	35.5	± 0.0	34.6	± 0.0
			1	34.1	± 0.4	33.5	± 0.1
			3	32.9	± 0.4	31.6	± 0.4
			5	31.9	± 0.3	29.9	± 0.4
		200 ms	0	35.6	± 0.0	-	-
			1	34.1	± 0.3	-	-
			3	32.8	± 0.3	-	-
			5	31.7	± 0.4	-	-
	144 kbit/s	50 ms	0	44.5	± 0.0	40.9	± 0.0
			1	40.0	± 0.3	39.7	± 0.2
			3	35.5	± 0.9	37.7	± 0.4
			5	34.6	± 0.5	36.2	± 0.5
		200 ms	0	44.5	± 0.0	-	-
			1	39.7	± 0.5	-	-
			3	36.0	± 0.8	-	-
			5	34.4	± 0.7	-	-
	384 kbit/s	50 ms	0	50.7	± 0.0	47.5	± 0.0
			1	39.5	± 0.9	45.9	± 0.3
			3	36.1	± 0.7	43.4	± 0.7
			5	34.3	± 0.6	41.7	± 0.5
		200 ms	0	50.7	± 0.0	-	-
			1	39.9	± 0.9	-	-
			3	35.2	± 0.8	-	-
			5	34.0	± 0.4	-	-
Foreman	144 kbit/s	50 ms	0	36.6	± 0.0	35.8	± 0.0
			1	35.4	± 0.1	34.3	± 0.2
			3	33.9	± 0.2	31.9	± 0.3
			5	32.3	± 0.1	30.0	± 0.4
		200 ms	0	36.5	± 0.0	-	-
			1	34.8	± 0.1	-	-
			3	32.6	± 0.1	-	-
			5	30.8	± 0.2	-	-
Soccer	144 kbit/s	50 ms	0	34.6	± 0.0	34.0	± 0.0
			1	33.5	± 0.1	32.9	± 0.1
			3	31.5	± 0.2	30.9	± 0.3
			5	30.0	± 0.2	29.2	± 0.3
		200 ms	0	34.6	± 0.0	-	-
			1	32.9	± 0.1	-	-
			3	30.3	± 0.2	-	-
			5	28.4	± 0.2	-	-

Table A.1: Calculated average Y-PSNR values and their corresponding 95% confidence interval.

APPENDIX B

Simulation details

This appendix provides details about the simulation parameters and how to reproduce the simulation process. The text assumes a Linux platform to run the modified JM software (found in the attached Zip-archive or online (see Appendix D)) and the testing environment described in Section 3.3.1.

B.1 Configuration parameters

This section gives a summary of the configuration parameters used in the simulations, both for the encoder and decoder. Parameters that are related to the specific set up of the testing environment are not described here, but as part of the simulation procedure described in Section B.2.

The implementation-specific parameters of RTP/AVPF are also listed in this section. These could be regarded as part of the implementation. However, since these are easily altered for new simulations (by changing the corresponding constants in the source code), they are considered as part of the configuration.

B.1.1 Encoder and decoder configuration

Table B.1 lists the encoder parameters that are common for the FBIR and IPLR schemes. Only the most important parameters are listed. The second block in the table contains parameters that are set for optimal coding speed. The third block lists parameters which affect the error robustness. In fact, most of the encoder configuration parameters are equal for the two schemes. The only two parameters that differ are those that enable FBIR and IPLR. For FBIR `FBIntraRefresh` was set to 2 (use FBIR with selective intra refresh), and for IPLR `IPLRMode` was set to 1.

Parameter	Value	Description
`ProfileIDC`	66	Baseline profile
`LevelIDC`	30	Level number 3
`OutFileMode`	2	Send as RTP stream to specified host
`NumberBFrames`	0	Disable B coded frames
`NumberReferenceFrames`	1	Store 1 frame in buffer for inter prediction
`UseFME`	2	Use fast motion estimation
`RDOptimization`	0	Disable RD-optimization
`RateControlEnable`	0	Disable rate control
`MbLineIntraUpdate`	0	No extra forced intra updates of GOBs
`RandomIntraMBRefresh`	0	No extra forced intra MBs per picture
`PartitionMode`	0	Disable partition mode
`num_slice_groups_minus1`	0	Disable FMO
`SliceMode`	1	Use a fixed number of MBs in each slice
`SliceArgument`	11	11 MBs (1 GOB) per slice
`UseConstrainedIntraPred`	1	Use constrained intra prediction

Table B.1: A selection of the encoder configuration parameters common for FBIR and IPLR.

			QP		Bit rate	
Sequence	Target rate	PLR	FBIR	IPLR	FBIR	IPLR
Conversation	64 kbit/s	0	28	29	61	63
		1	28	29	62	63
		3	28	29	63	63
		5	28	29	64	63
	144 kbit/s	0	17	21	143	141
		1	17	21	144	141
		3	17	21	146	141
		5	17	21	149	141
	384 kbit/s	0	9	13	361	364
		1	9	13	364	364
		3	9	13	365	364
		5	9	13	367	364
Foreman	144 kbit/s	0	26	27	143	152
		1	26	27	146	152
		3	26	27	151	152
		5	27	27	142	152
Soccer	144 kbit/s	0	28	29	139	145
		1	28	29	140	145
		3	28	29	142	145
		5	28	29	143	145

Table B.2: All quantization parameters and the resulting bit rates for each sequence and parameter set.

Table B.2 lists the applied quantization parameters and the resulting bit rates, including IP/UDP/RTP-packet overhead. Equal quantization was used for both I and P slices. Naturally, there are small differences in the bit rates, but the differences are so small that the obtained results are comparable. The bit rates listed for the FBIR scheme are with 50 ms network latency. For 200 ms latency there are normally a slight increase in rate, in the range 0–2 kbit/s.

For the decoder there are few parameters that are accessible through the configuration file. All values were kept at their default values, except from "NAL mode" which was set to 2 in order to receive RTP packets from the network, and the error concealment strategy which was set to 1 ("motion copy"). A parameter to indicate the RTP session bandwidth was made available through the implementation and set to 64, 144, or 384, depending on the encoded bit rate.

B.1.2 RTP/AVPF parameters

Since RTP/AVPF has many application-specific variables, the temporal behaviour of the feedback scheme will depend on the implementation. Table B.3 lists the values for all variables and constants used in this study that affect the timing of the feedback. These are only relevant for the FBIR scheme, of course. For simplicity, some variables were set to a fixed value, such as the number of senders and receiver. These are also listed. For convenience, the table uses the same variable names as in the standards [4] and [5]. Except from `rtp_bw` which is accessible through the decoder configuration file, all parameters are set in the source code. See Appendix C for details on how to access these.

Parameter	Value	Description
`senders`	1	Number of senders in RTP session
`receivers`	1	Number of receivers in RTP session
`T_dither_max`	150 ms	Max additional delay for FB
`T_max_fb_delay`	∞	Upper bound for FB to be useful
`T_rr_interval`	5 sec	Minimum interval between regular RTCP packets with no FB message
`rtcp_bw`	0.05 ∗ `rtp_bw`	Total available RTCP bandwidth for all members
`rtp_bw`	64, 144 or 384 kbit/s	Applied RTP session bandwidth
`rtp_max_delay`	50 ms	Max delay before an RTP packet is considered lost (not in standard)

Table B.3: Implementation-specific decoder parameters.

B.2 The procedure

First, this section explains the process of encoding and decoding a test sequence using the setup from Section 3.3.1. In short, this implies real-time encoding and decoding, where the encoded video is transmitted to the decoder through a network emulator. Then, it is explained how to measure the objective quality for the received video.

Setting/parameter	Value
Test sequence	Foreman
Bit rate	144 kbit/s
Encoder host	10.0.0.3
Decoder host	10.1.1.3
Encoder port, incoming RTCP	62003
Decoder port, incoming RTP	62002
Packet loss rate	1%
Network latency	50 ms
Network emulator address	packetsphere.item.ntnu.no

Table B.4: Settings/parameters specific for each simulation and/or testing environment setup.

The procedure is explained only for one parameter set since the steps are equivalent for other parameter sets, but with other input parameters of course. The example will simulate transmission of the Foreman sequence at 144 kbit/s over a network with 1% packet loss rate and 50 ms latency. All settings/parameters that are specific to this particular example and test bed setup are listed in Table B.4.

B.2.1 Prepare the software

The modified JM software is provided only as source code. In order compile the software into executable files, the tools *make* and *gcc* are required. First, get the modified JM source code, either from the attached Zip-file or the online archive (see Appendix D). After extracting it, a directory called `source/jm` is created. The encoder is found in the sub-directory `source/jm/lencod` and the decoder in `source/jm/ldecod`. The encoder and decoder are provided with a Makefile each, which lies in the respective directories. In order to compile the encoder or decoder, execute the following commands

```
cd source/jm/lencod; make; cd -
```

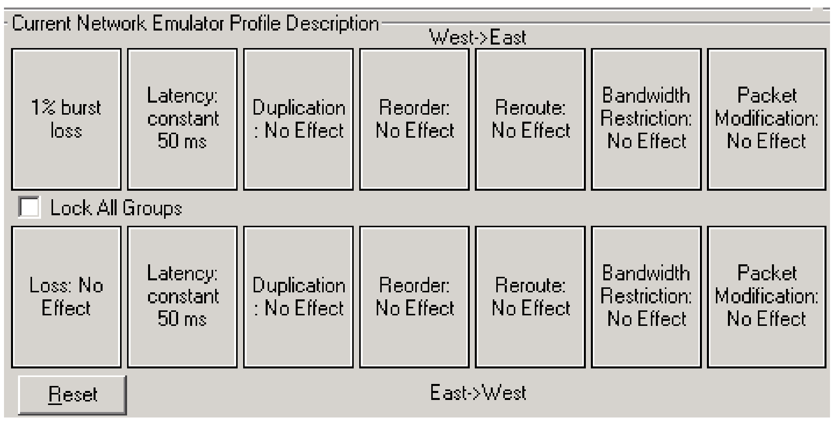
or for the decoder

```
cd source/jm/ldecod; make; cd -
```

This will create executable files in `source/jm/bin`. The encoder and decoder are named `lencod.exe` and `ldecod.exe`, respectively. They should be compiled on two separate machines connected through a LAN with the network emulator in the middle, according to the test bed setup. The encoder was installed on `kurt.q2s.ntnu.no` and the decoder on `manage.q2s.ntnu.no`.

EvalVid's PSNR program is installed with the same approach. The source code for this program is found in the same Zip-archive as the codec, described above. After extraction there should be a directory named `source/psnr`. It is only necessary to compile the PSNR program at the machine where the decoder was compiled:

```
cd source/psnr; make; cd -
```



Current Network Emulator Profile Description

West->East

1% burst loss	Latency: constant 50 ms	Duplication : No Effect	Reorder: No Effect	Reroute: No Effect	Bandwidth Restriction: No Effect	Packet Modification: No Effect

Lock All Groups

Loss: No Effect	Latency: constant 50 ms	Duplication : No Effect	Reorder: No Effect	Reroute: No Effect	Bandwidth Restriction: No Effect	Packet Modification: No Effect

East->West

Figure B.1: PacketSphere Network Emulator configured to simulate a 1% loss rate with a 50 ms latency.

PacketSphere is used as the network emulator. The emulator is controlled by a software application which may be downloaded and installed from `http://packetsphere.item.ntnu.no/downloads.asp`. This is a Windows-only program. The usage of this program is very intuitive and will not be explained in detail here. The main steps to connect and set up the emulator are to first to add the emulator network address (`packetsphere.item.ntnu.no`) to the list of PacketSphere Servers. Then, resources have to be reserved before the emulator is started. Network parameters can be altered while the emulator is running. (It will restart automatically.)

B.2.2 The simulation

In order to run the simulation, the emulator has to be started with the correct parameters, the packet capture device needs to be activated, and the decoder and encoder programs must be started with the correct configuration.

First, set up PacketSphere to randomly loose 1% of the packets transmitted from the encoder to the decoder. The losses should be set to occur in bursts of at least one packet, but no more than three. The back-channel from the decoder to the encoder should be error free. Both the forward and backward channel should be set to have a 50 ms constant latency. The resulting status of the network emulator is shown in Figure B.1.

Second, the network monitoring interface card has to be activated. However, this is optional as the captured packets only are used for analyzes of the network flow and packet formats. The network protocol analyzer Wireshark [41] may be used for this purpose. Since this is not necessary in order to measure the video quality, the usage of the captured packets will not be explained any further. The DAG card used in the simulations was pre-installed on `q2s.uninett.ntnu.no` and activated by:

```
dagsnap -d /dev/dag1 -v -o capture.trace
```

The decoder must be started before the encoder because the decoder is able to listen for incoming RTP packets, while the encoder starts the transmission immediately. A default configuration file is provided along with the decoder. It is named `decoder_rtp.cfg` and is found in the directory `source/jm/bin`. This file must be modified according to the current test bed setup and the RTP session bandwidth. The parameters that may need to be adjusted are the incoming RTP port, the encoder's IP-address, the encoder's RTCP port, and the bit rate. The decoder will both save the compressed bit stream received from the decoder and the decoded uncompressed video. The default file names are `received.264` and `received_dec.yuv`, respectively, but can be changed in the configuration file. Finally, to start the decoder and save the output log to `dec.log` (optional, but is needed for inspection of RTP/AVPF's temporal properties), run the commands

```
cd source/jm/bin
./ldecod.exe decoder_rtp.cfg > dec.log
```

The decoder will quit if it does not receive any data before 10 seconds. The commands above are performed for both FBIR and IPLR. This implies that for the decoder will send feedback both for FBIR and IPLR, but when the encoder is in IPLR mode the feedback will be ignored.

The encoder, on the other hand, must be configured according to the applied ER scheme. A default configuration file, `encoder.cfg` is provided for the encoder in the `source/jm/bin` directory. All encoding parameters that are common for FBIR and IPLR are set as default. In contrast to the decoder, the encoder configuration file does not need direct modification. Instead, the parameters are easily altered from the command line. However, it could be convenient to change the configuration file if several simulations are performed in order to save some typing. The parameters are set according to Table B.2 and B.4. Because of storage size considerations, the attached/downloaded Zip-archive contains only the Foreman sequence with 300 frames. This file will be used in this example. The full length test sequences used in the simulations can be downloaded from the online archive (see Appendix D). The command to start encoding the Foreman sequence at 144 kbit/s with the FBIR scheme is

```
cd source/jm/bin
./lencod.exe -d encoder.cfg \
    -p InputFile="foreman_qcif_10fps_300f.yuv" \
    -p ReceiverHost=10.1.1.3 -p ReceiverPort=62002 \
    -p FeedbackPort=62003 \
    -p QPISlice=26 -p QPPSlice=26 \
    -p FBIntraRefresh=2 -p IPLRMode=0
```

The corresponding command for IPLR is

```
cd source/jm/bin
./lencod.exe -d encoder.cfg \
    -p InputFile="foreman_qcif_10fps_300f.yuv" \
    -p ReceiverHost=10.1.1.3 -p ReceiverPort=62002 \
    -p FeedbackPort=62003 \
    -p QPISlice=27 -p QPPSlice=27 \
    -p FBIntraRefresh=0 -p IPLRMode=1
```

The decoder should now be decoding the file transmitted from the encoder. However, if some of the packets containing the sequence or picture parameter set are lost, the decoder may fail and the process must be restarted. To check the progress of the decoding, inspect the output of the decoder which is being dumped to `dec.log`.

```
tail -f dec.log
```

After successfully decoding the transmitted video, several files have been generated. The encoder logs information about the FBIR and IPLR algorithms in `fbir.log` and `iplr.log`, respectively. These may be used to inspect the detailed intra refresh behaviour for the encoder. Most important are the files generated by the decoder. `received.264` is the received packet stream, which may be decoded (again) offline at a later time. `received_dec.yuv` is the decoded video. Finally, `dec.log` contains information about the decoding and feedback process.

B.2.3 Measure the objective quality

In order to measure the objective quality for the decoded video, the PSNR program is used to calculate the Y-PSNR value with the original video file as reference. Thus, the decoder must also have a copy of the original yuv-file. To easily calculate the Y-PSNR and save the results to a file name `received.psnr`, run the program while still having `source/jm/bin` as the current working directory:

```
../../psnr/psnr 176 144 420 foreman_qcif_10fps_300f.yuv \
    received_dec.yuv > received.psnr
```

B.2.4 Perform numerous simulations

The simulation procedure explained thus far generates one Y-PSNR measure for one parameter set and sequence. For this study there were 5 or 10 simulations for each parameter set. Therefore, some of the steps above was automated to save time. Unfortunately, there is no easy way to synchronize and control the encoder, decoder, network emulator, and network interface card. Thus, the encoding/decoding process must be executed manually for each simulation. However, the calculation of Y-PSNR is easy to automate.

The total simulation process carried out can be outlined as follows: First, all encoding/decoding for all parameters sets were performed. The received video (`received.264`) and generated log-files were stored according to a defined directory structure. The decoded video `received_dec.yuv` was discarded since it could be generated from `received.264` at a later time. Second, an ad hoc script were developed to calculate the Y-PSNR for all video files. This script iterated through all simulations and decoded the compressed video, calculated the Y-PSNR, and finally deleted the uncompressed video to save storage space. Finally, other ad hoc scripts were created to extract and plot different sorts of information from the log-files. The attached Zip-archive contains some of these scripts, but they are not runnable since they require a specific directory structure and/or Python modules.

B.2.5 Alternative approach to test the encoder/decoder

The simulation procedure described so far requires a testing environment that is not widely accessible. There exists, however, an alternative approach to perform real-time encoding and decoding with FBIR and IPLR. That is to

encode and transmit the packet over a "normal" network or through the loop-back interface on the computer. Hence, the decoder may be running on another machine or the same machine as the encoder. The latter will be assumed in the following. A powerful processor is required. This approach does not simulate network delay, but there exists a simple network simulator in the modified encoder, which is able to drop packets according to the Gilbert model [3]. The encoder and decoder host must be set to 127.0.0.1 and the Gilbert network simulator must be enabled. Besides this the procedure is mostly the same as described before. The decoder command will remain unchanged (but the configuration file must be changed). To simulate a packet loss rate of 1% with an expected burst length of 2 packets, the encoder commands for FBIR become

```
cd source/jm/bin
./lencod.exe -d encoder.cfg \
    -p InputFile="foreman_qcif_10fps_300f.yuv" \
    -p ReceiverHost=127.0.0.1 -p ReceiverPort=62002 \
    -p FeedbackPort=62003 \
    -p QPISlice=26 -p QPPSlice=26 \
    -p FBIntraRefresh=2 -p IPLRMode=0 \
    -p GilbertEnable=1 -p \
    -p GilbertLossRate=1 -p GilbertBurstLength=2
```

For IPLR, just change the `FBIntraRefresh` and `IPLRMode` parameters.

Since this procedure does not introduce any network latency, the encoder will receive feedback messages earlier and the performance of FBIR will increase. However, this is accepted since this approach only should be used to test the concept of the ER schemes, not their performance.

APPENDIX C

Implementation details

As mentioned in Section 3.2, the H.264 reference coder JM 11.0 was modified to support FBIR and IPLR. This appendix provides an overview over main modifications in the software and where to find the relevant parts of the ER schemes. The intention is not to give a description of the code itself since the reader may download and inspect the source code for himself (see Appendix D). The coder is found in the directory `source/jm`. For details on compilation, see Section B.2.1.

All modifications that affect the default behaviour of JM 11.0 is enclosed by the preprocessor directives `#if` and `#endif` for conditional inclusion. Hence, it is easy to extract the code written as part of this thesis from the rest. In addition, it is easy to roll back to default JM behaviour by defining all macros that enable the implemented features as 0. As an example, consider the following code

```
#if FBIR
  if (input->FBIntraRefresh != 0)
    OpenRTCPConnection(input->feedbackPort);
#endif
```

which only is included in the compilation if the macro `FBIR` is different from 0. All macros that enables/disables added features are defined in `defines.h`, for both the encoder and decoder.

C.1 The encoder

Many files are modified in the encoder. The most important files relevant for the ER schemes described in Section 2 are `lencod.c`, `mode_decision.c`, `rtp.c`, `rtcp.c`, `fbir.c` and `iplr.c`, in addition to the header file `defines.h`. These files were created/modified in order to obtain the following:

`defines.h`: Modified to define the macros that enable/disable the added features. These are `FBIR`, `IPLR`, `RTP_OVER_NETWORK`, `RTP_BIG_ENDIAN_SUPPORT` and `GILBERT_IN_ENCODER`. (Not all are relevant for the functionality described here.)

`lencod.c`: Modified to force the encoder to encode in real-time (if the processor is fast enough), i.e not encode more than the given frames per second. Real-time encoding is enabled only in the cases where the encoder is configured to transmit the encoded video over the network.

`mode_decision.c`: Modified to check if intra refresh should be performed for each MB, both for FBIR and IPLR, and enforce intra coding if it should. The code in this file is also responsible for updating the variables for "near static" detection.

`rtp.c`: Modified to send RTP packets over the network to a given receiver according to configuration parameters.

`rtcp.c`: Added to listen for incoming RTCP packets on a specified port. When packets arrive they are decomposed and interpreted, and other parts of the encoder are notified.

`fbir.c`: Added to do perform the main parts of the FBIR scheme, except from the RTP/AVPF routines. This includes error tracking, to store MVs used in the error tracking algorithm, and to store information about packets reported lost.

`iplr.c`: Added to run the IPLR algorithm.

For further information the reader is referred to the source code itself.

C.2 The decoder

The modifications in the decoder add fewer additional features compared with the encoder, but the features added are more complex and very important. The files of interest are `defines.h`, `rtp.c` and `rtcp.c`. These files were created/-modified in order to obtain the following:

`defines.h`: Modified to define the macros that enable/disable the added features. These are `RTP_OVER_NETWORK`, `RTCP_FEEDBACK` and `RTP_BIG_ENDIAN-_SUPPORT`.

`rtp.c`: Modified to receive RTP packets from the network on a specified port. A reception buffer is implemented so that packets are allowed to be out of sequence. A packet is considered lost when it is not received before an expected time limit.

`rtcp.c/rtcp.h`: Added to create, compose and send RTCP packets according to RTP/AVPF. This file contains all the transmission rules, packet formatting etc.. All RTP/AVPF implementation-specific parameters are defined in the header file.

For further information the reader is referred to the source code itself.

APPENDIX D

Archives

Archives with the source code and some video examples are provided for the reader. This Appendix describes the contents of the attached Zip-archive and the more extensive archive found online.

D.1 Attached Zip-archive

The Zip-archive submitted together with the thesis contains the source code for the modified JM coder, the Y-PSNR program from EvalVid, some ad hoc scripts used in the simulation process and finally, some video examples. For a more detailed description the reader is referred to the file `README.txt` found in the root directory of the archive.

The video examples are in the mp4-file format. This is done to make it easy to view the videos, at least for Windows and Mac users. The files may be viewed with Apple's QuickTime [42]. The decoded video will not look exactly the same as it would if decoded with the JM decoder, because QuickTime employs another (and slightly improved) error concealment scheme. However, the visual appearance are very close.

The provided videos are Conversation at 64 kbit/s with a 1% packet loss and 50 and 200 ms latency, both for the FBIR and IPLR scheme; and Foreman at 144 kbit/s with a 3% packet loss and 50 and 200 ms latency for both schemes.

D.2 Online archive

The online archive [43] contains a comprehensive set of video examples, one example for each combination of ER scheme, parameter set and test sequence.

It also contains the Zip-file described above where the source code may be downloaded, and the full length test sequences used in the simulations.

The online archive is found at

```
http://www.pvv.ntnu.no/~stianse/master
```

The get access you must enter the following username and password:

```
username: master
password: thesis
```

For a more detailed description on the usage of this archive the reader is referred to the file `readme.txt` found in the root directory of the archive.