TECHNISCHE UNIVERSITÄT WIEN

VIENNA UNIVERSITY OF TECHNOLOGY

TREBALL DE FI DE CARRERA

TÍTOL: Improved error detection in H.264 encoded video stream for mobile networks

AUTOR: Jacob Cañadas Rodríguez

SUPERVISORA: Olívia Némethová

PROFESSOR: Markus Rupp

DATA: 24 of January 2004

Títol: Improved error detection in H.264 encoded video stream for mobile networks

Autor: Jacob Cañadas Rodríguez

Titulació: Enginyeria Tècnica de Telecomunicació

Especialitat: Telemàtica

Pla: 2000

Supervisora: Olívia Némethová

Profesor: Markus Rupp

Departament: Institute for Communications and Radio Frequency Engineering, University of Technology Vienna, Austria

País: Austria

Vist i plau,

Director del TFC

Registre:

Títol: Improved error detection in H.264 encoded video stream for mobile networks

Autor: Jacob Cañadas Rodríguez

Supervisor: Olívia Némethová

Professor: Markus Rupp

Data: 10 of January 2004

Resum

En aquest treball trobarem un estudi del codec H.264. Aquest nou codec incorpora nous mecanismes per tal de donar una millor compressió i qualitat comparat amb els seus antecessors (MPEG-4,H.263). Aquest treball prova de optimitzar les propietats d'aquest codec per tal d'utilitzar-lo a xarxes mòbils.

En el primer capítol s'exposaran conceptes bàsics sobre el codec H.264 per tal de posar les bases de coneixement per tal d'entendre correctament el treball. Més tard al segon capítol es parlarà de la problemàtica d'utilitzar aquest codec en entorns wireless.

Amb aquests coneixements al capítol 4 s'utilitzaran mecanismes del propi codec per tal de millorar les prestacions d'aquest. Mentre que al capítol 5 es farà el mateix però utilitzant nous mecanismes .

Finalment a l'últim capítol es farà una conclusió final proposant la utilització d'alguns mètodes dels que han estat proposats durant el treball.

Title: Improved error detection in H.264 encoded video stream for mobile networks

Author: Jacob Cañadas Rodríguez

Supervisor: Olívia Némethová

Profesor: Markus Rupp

Date: 10 of January 2004

Overview

In this project we will find a study about H.264. This new codec includes new properties for having better compression and quality than the old codecs (MPEG-4, H.263). This work tries to improve the error detection in H.264 for mobile networks.

In the first chapter there is an overview of H.264 codec for having the necessary knowledge to understand correctly the project. Then in the next chapter we will talk about the problematic of using H.264 in mobile networks.

With this knowledge in the chapter 4 we will use error resilience mechanisms of the H.264 codec to try to improve the quality of it. Furthermore, in the chapter 5 we will try to make the same but using news mechanisms not included in the codec.

Finally in the last chapter we will make a final conclusion and we will propose the usage of chosen methods to improve the error detection in mobile networks.

ÍNDEX

SECTION 1. INTRODUCTION

H.264 is a newest codec that try to improve the properties of h.263 and MPEG-4. This codec gives better quality with less bandwidth than the others. This is very interesting for mobile networks because in these environments the bandwidth is so restrictive.

But not all is good, like other codecs, H.264 has problems with errors. Detection, propagation and concealment are some of the things that H.264 has problems on.

In this work we have study a lot of proposed methods for improve the properties of H.264, there is no global solutions. There are partial solutions focussed in some specific examples.

In this project we have take this methods and we have study it for be able to have a global perception. First we have focus in error resilience features of H.264. Slicing, intra-update, data partitioning, etc are very important for the final efficiency of the codec.

With the features of H.264 is not sufficient for taking good results in mobile networks. There are several problems with the detection and propagation of the errors. For mend this in the section 5 are exposed several additional methods. These methods are new and some modifications of the codec are needed to use it.

Synchronization marks, out-of-band and parity bits try to improve the granularity of the error detection. H.264 uses VLC (Variable Length Codes) codes to encode the bitstream; these codes give a good compression but have problems with errors. The new methods proposed minimized the impact in case of errors.

Then in the final chapter a final method is proposed taking the best methods project for mobile networks.

SECTION 2. OVERVIEW OF H.264

2.1 Overview of H.264/AVC

This is a newest video coding standard of the ITU-T Coding Experts Group and the ISO/IEC Moving Picture Group. The main goals of the H.264/AVC standardization effort have been enhanced compression performance and provision of a "network friendly" video representation addressing "conversational" (video telephony) and "nonconversational" (storage) applications. H.264/AVC has achieved a significant improvement in rate-distortion efficiency relative to existing standards.

The new standard is designed for technical solutions including different areas like: Broadcast, Storage, Conversational, Streaming, Multimedia messaging services, etc. H.264 handles this variety of applications and networks using the Video Coding Layer (VCL) that is designed to efficiently represent the video content and de Network Abstraction Layer (NAL) which formats the VCL representation of the video and provides header information in a manner appropriate for conveyance by a variety of transport layer or storage media. In Figure 2.1 there are the structure of H.264 video encoder.

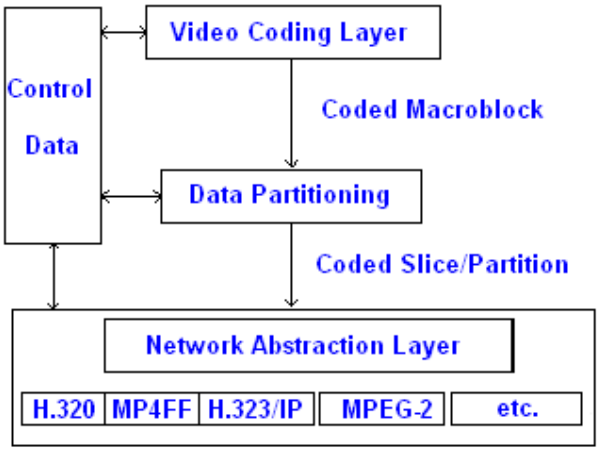


Figure 2.1 Structure of H.264/AVC video encoder

2.1.1 NAL units

The NAL is designed in order to provide "network friendliness" to enable simple and effective customization of the use of the VCL for a broad variety of systems. The NAL facilitates the ability to map H.264/AVC VCL data to transport layers such as:

- RTP/IP: Real-time internet services
- File formats: e.g., ISO MP4 for storage and MMS.
- H.32X: wireline and wireless conversational services.
- MPEG-2: Broadcasting.

NAL units are classified into VCL and non-VCL NAL units. The VCL NAL units contain the data that represents the values of the samples in the video pictures, and the non-VCL NAL units contain any associated additional information such as parameter sets and supplemental enhancement information.

The parameter set is supposed to contain information that is expected to rarely change and offers the decoding of a large number of VLC NAL units.

2.1.2 VCL

In the VCL each coded picture is represented in block-shaped units of associated luma and chroma samples called macroblocks. There is no sinble coding element in the VCL that provides the majority of the significant improvement in compression efficiency in relation to prior video coding standards. It is rather a plurality of smaller improvements that add up to the significant gain.

For more details see [1].

2.2 Prediction Modes

The basic source-coding algorithm is a hybrid of inter-picture prediction to exploit temporal statistical dependencies and transform coding (intra-prediction) of the prediction residual to exploit spatial statistical dependencies.

2.2.1 Intra prediction

If a block or macroblock is encoded in intra mode, a prediction block is formed based on previously encoded and reconstructed blocks. This prediction block P is susbstracted form the current block prior to encoding. For the luminance samples, P may be formed for each 4x4 luminance block; 4 optimal modes for 16x16 luminance block; and one mode that is always applied to each 4x4 chroma block.

In this case only spatial references are used. This is the prediction mode for Intra-frames.

2.2.2 Inter prediction

Inter prediction creates a prediction model from one or more previously encoded video frames. The model is formed by shifting samples in the reference frame(s) so in this case this is a temporal prediction. This is the prediction used for P/B frames.

For more information read [2],[3].

2.3 Transform and Quantization

Each residual macroblock is transformed, quantized and coded. The “baseline” profile of H.264 uses three transforms depending on the type of residual data that is to be coded: a transform for the 4x4 array of luma DC coefficients in intra macroblocks, a transform for the 2x2 array of chroma DC coefficients and a transform for al other 4x4 blocks in the residual data.

2.3.1 Transform

For example the transform operates on 4x4 blocks of residual data after motion-compensated prediction or Intra prediction. The transform is based on the DCT but with some differences: Is is an integer transform, the inverse transform is fully specified in the H.264 standard, etc.

2.3.2 Quantization

H.264 uses a scalar quantizer. The definition and implementation are complicated by the requirements to avoid division and/or floating point arithmetic and incorporate the post- and pre-scaling matrices.

The basic forward quantizer operation is as follows:

$$Z_{ij} = round\,(Y_{ij} / Qstep) \qquad \textbf{(2.1)}$$

Where Yij is a coefficient of the transform described above, Qstep is a quantizer step size and Zij is a quantized coefficient.

A total of 52 values of Qstep are supported by the standard and these are indexed by a Quantization Parameter, QP. Qstep increases by 12.5% for each increment of 1 in QP. The wide range of quantizer step sizes makes it possible

for an encoder to accurately and flexibly control the trade-off between bit rate and quality.

For more information read [4].

2.4 Variable Length Coding

There are two types of entropy coding specified in the standard:

- Context-based Adaptive Arithmetic Coding (CABAC)
- Variable-Lenght Coding (VLC)

At the slice layer and below, elements are encoded as fixed- or variable-length codes (VLCs) or context-adaptive arithmetic coding (CABAC) depending on the entropy encoding mode. CABAC it's not so much robust for mobile networks so this document will be focussed in VLC codes.

When entropy_coding_mode is set to 0, residual block data is coded using a context-adaptive variable length coding (CAVLC) scheme and other variable-length coded units are coded using Exp-Golomb codes.

2.4.1 Exp-Golomb entropy coding

Exp-Golomb codes are variable lenght codes with a regular construction. In Figure 2.1 is possible to see the table of this method.

Table 2.1 Bit string table taked from the standard.

Bit string form	Range of Code Num
1	0
0 1 x_0	1-2
0 0 1 x_1 x_0	3-6
0 0 0 1 x_2 x_1 x_0	7-14
0 0 0 0 1 x_3 x_2 x_1 x_0	15-30
0 0 0 0 0 1 x_4 x_3 x_2 x_1 x_0	31-62
…	…

The codewords progress in a logical order [6]. Each codeword is constructed as follows:

[M zeros][1][INFO]

Where INFO is an M-bit field carrying information. The first codeword has no leading zero or trailing INFO; codewords 1 and 2 have a single-bit INFO field; codewords 3-6 have a 2-bit INFO field; and so on. The lenght of each codeword is (2M+1) bits.

Exp-Golomb is designed to produce short codewords for frequently-ocurring values and longer codewords for less common parameter values.

2.4.2 Context-based adaptive variable length coding (CAVLC)

This is the method used to encode residual, zig-zag ordered 4x4 (and 2x2) blocks of transform coefficients. CAVLC is designed to take advantage of several characteristics of quantizaed 4x4 blocks:

- After prediction, transformation and quantization, blocks are typically sparse (containing mostly zeros). CAVLC uses run-level coding to compactly represent strings of zeros.
- The highest non-zero coefficients after the zig-zag scan are often sequences of +/- 1. CAVLC signals the number of high-frequency +/-1 coefficients in a compact way.
- The number of non-zero coefficients in neighbouring blocks is correlated. The number of coefficients using a look-up table; the choice of look-up table depends on the number of non-zero coefficients in neighbouring blocks.
- The level of non-zero coefficients tends to be higher at the start of the reordered array by adapting the choice of VLC look-up table for the “level” parameter depending on recently-coded level magnitudes.

CAVLC encoding of a block of transform coefficients proceeds as follows:

1. Encode the number of coefficients and trailing ones.
2. Encode the sign of each trailing +/- 1.
3. Encode the levels of the remaining non-zero coefficients.
4. Encode the total number of zeros before the last coefficient.
5. Encode each run of zeros.

SECTION 3. MOBILE NETWORKS AND ERROR PROPAGATION

3.1 Introduction

Video transmission for mobile terminals is likely to be a major application in emerging 3G systems and may be a key factor in their success. The video-capable display on mobile devices paves the road to several new applications.

In general, the available bandwidth and therefore the bit-rate over the radio link are limited and the costs for a user are expected to be proportional to the reserved bit rate or the number of transmitted bits over the radio link. Thus, low bit rates are likely to be typical, and compression efficiency is the main requirement for a video coding standard to be successful in a mobile environment.

In addition, the mobile environment is characterized by harsh transmission conditions in terms of attenuation, shadowing, fading, and multi-user interference, which result in time- and location-varying channel conditions. The frequency of the channel variations highly depends on the environment, the user topology, the velocity of the mobile user, and the carrier frequency of the signal.

3.2 Transport of H.264/AVC Video in Wireless Systems

In the Figure 3.1 is possible to see one scenario in which one Streaming Server (SS) is transmitting video over the core network (CN) to the mobile system (Ina a UMTS environment).

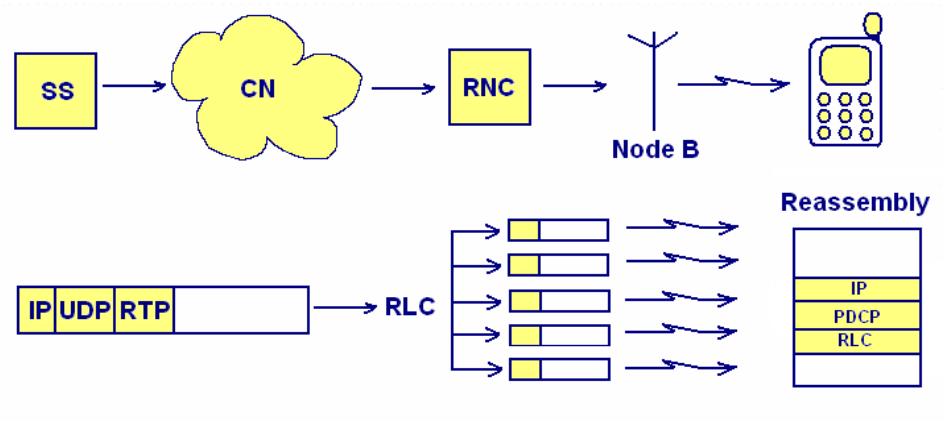


Figure 3.1 Structure of streaming transmission.

The streaming server send one IP packet over the network, this packet in the Radio Network Control is splitted into several RLC-PDU packets (for the common UMTS networks the size of this RLC packets is 320 bits). The mobile receive all this packets. It's possible to see that the mobile in his protocol stack has IP upper RLC.

So when the mobile receive the packets he has to merge the RLC packets to have the IP packet. The problem is if one RLC packet has an invalid CRC, this packet will be discarded so when the entire RLC packet will be merged into the IP packet, this will have the CRC wrong. The mobile is not able to know where in the IP packet is the error so it's needed to discard entire one.

Like read in the second section, H.264 uses temporal references so if this packet is lost the error will be propagated in the time until de next synchronization point.

Really there were only 320 bits wrong but this error had been propagated over the entire IP packet (size until 1500bytes), and not only this, the error had been propagated over the time until the next synchronization point.

It's necessary to use some system to mend this problem, because the granularity of the codec is so small. This makes that the H.264 will be not efficient in mobile networks.

SECTION 4. ERROR RESILIENCE FEATURES OF H.264

4.1. Intra-update

H.264 allows inserting some I macroblocks or whole frames into the encoded sequence. Inserting I macroblocks helps to refresh the picture, but cannot help in case of bigger artifacts that are likely to occure in environments with lower SNR. Therefore regular inserting of I frames is an important feature to stop the error propagation. Drawback of this method is increased data rate, which is also critical for mobile communication where usually a dedicated channel is allocated to the user with a fixed data rate. Thus, it is important to find a tradeoff between the error propagation consequences and the overhead of I frames.

4.1.1. Overhead

To see the overhead introduced by using I frames three frames of the foreman video were taken and encoded in an error-free environment in two ways:

- 1.-With only the first frame INTRA predicted (IDR). So the sequence will be IDR-P-B
- 2.-With one more I frame. In this case the sequence will be IDR-I-B

The results of the test are in the Table 4.1.

Table 4.1 Results of the test.

	Encoding time (sec)	SNR Y (dB)	SNR U (dB)	SNR V (dB)	Total bits (bits)	Bit rate (kbps)
1(No Intra)	41,64	36,7	41,03	42,54	32272	322,72
2 (Intra)	15,03	37	41,24	42,88	46104	461,04

First of all it is necessary to say that the first frame in every sequence has an IDR frame that is intra predicted, also in the two cases the last frame is codec like an B frame that is inter-predicted. So the only difference is that the second frame in one case is an I-frame and in the other is a P-frame. Also at the beginning there is a packet with the information about the parameter sets that is for the two cases the same.

In the Table 4.1 is possible to see that make intra-prediction take less time to encode. This is because with intra-prediction don't have references from other frames so it can be encoded alone only making spatial prediction using the

neighbors. In the other hand it needs more bits because it does not use references to another[jcr1] frames, I frames are defined by themselves.

4.1.2. Problems of Intra prediction

The problem is that INTRA adds so much overhead. Normally P packets sizes are only 10% of the length of one I packet if encoded in slicing mode 0.

In constant bit rate (CBR) coding, I MBs or slices lead to a significantly decreased image quality.

Intra prediction is good to improve the quality of the image but the best issue of this prediction is to stop the temporal propagation of the errors. The problem this carries a lot of overhead so it is not a good idea to abuse this method.

4.1.3. Possible solutions

With H.264 there is the possibility to introduce intra macro blocks, slices or frames. So it can be chosen the level in which Intra information wants to be added.

It was exposed that increases the frequency of intra coded frames is not a viable solution to stop the temporal propagation of the errors. But with H.264 is possible to make only macro blocks to be intra-predicted inside a P/B frame.

So is important to calculate the optimum number of INTRA MBs per frame. There are different ways to do it:

- Periodical INTRA refresh of video frames
- Periodical update of randomly chosen MBs.
- Rate distortion approach.

Without being careful this methods cannot be implemented in real-time environments. They add too much complexity.

The same test as in previous section will be used to estimate the aproppiate number of Intra-MBs that can be added according to the overhead in front of P-MBs. In the test there is only one slice per frame, in the next table it is possible to see the bits needed for a MB in each case (Every frame have 16x16 MBs):

Table 4.2 Size of I and P frames in the test.

	Bits for a frame (bits)	Bits for a MB
I frame	21912	86
P frame	8304	33

In the Figure. 4.1 it is possible to see the overhead for different number of Intra-MB added in a P-frame.

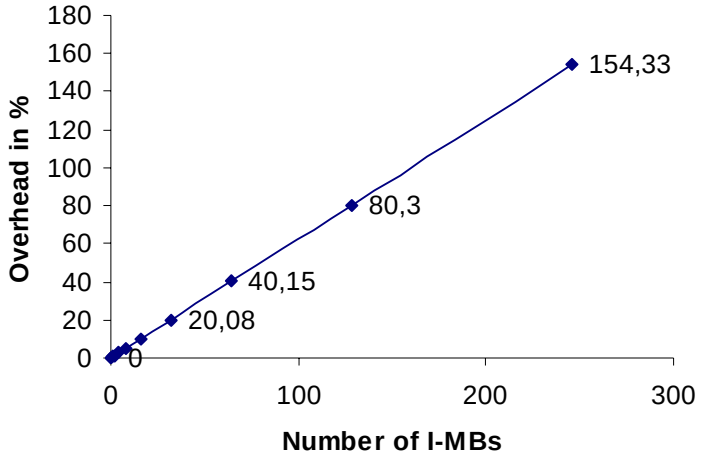


Figure 4.1 Overhead for different number of I-MBs.

It is possible to see that adding more than 20 I-MBs for each frame is not viable so this graphic will be focused in values under 20 MBs for frame. Also normally 2-10 MBs are used when encoding with H.264, so the focus will be in the fewer values.

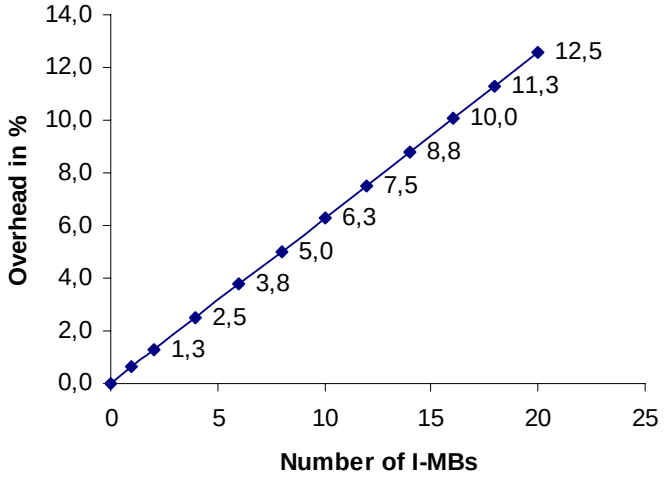


Figure 4.2 Overhead for different number of I-MBs.

To choose the correct number of I-MBs it is necessary to see this table and study the channel in witch the sequence will be transmitted. Another question is where to add this I-MBs; at the beginning, end, randomly, other particular place? Depending of the environment there are two interesting methods.

- Randomly chosen
- Update regions with much motion

In the first method it is only needed to specify the number of I-MBs per frame/slice than have to be added in every temporal-predictive frame (P/B). It's not a good idea to choose temporally adjacent MBs at the same spatial location because the coding gain will be considerably reduced. This is why randomly choose will be a possible solution.

Normally there are regions in the frame that are more important than other, for example, in a football video is more important to see where is the ball and the players that are running nearly than to see the lawn with a high quality.

The second method appears for this purpose. Like it is possible to read in [10], it is possible to Intra code only the MBs with much motion repeatedly. This method will add the same overhead than the first one but the result will be a better quality perception for the end-user. The unique problem is to calculate where are these regions with more motion. So this method give more quality with the same overhead but with more complexity, depending of the environment will have to be used one or the other.

4.1.4. Conclusions

For this method is proposed to use the Figure 4.2 and choose the correct value of I-MBs depending of the possibility to add overhead and add it in the random way. This is one method that will not add so much complexity to the encoder.

4.2. Slicing

Packet loss probability and the visual degradation from packet losses can be reduced by introducing slice-structured coding.

Each frame is subdivided into MBs. Slice is a group of MBs, that provides spatially distinct resynchronization points within the video data for a single frame. If encoded as an RTP stream, one slice is usually encapsulated into one RTP packet without segmentation or assembly.

The easiest way to form the slices is to have one frame per slice. This method is simple, but not efficient. Efficient and frequently used method is to subdivide the frame into the slices with the same number of bits. This is especially well

suited for real-time communication over packet networks as it leads to the packets with the same size.

In the Figure 4.3, other three possible slicing methods can be found.

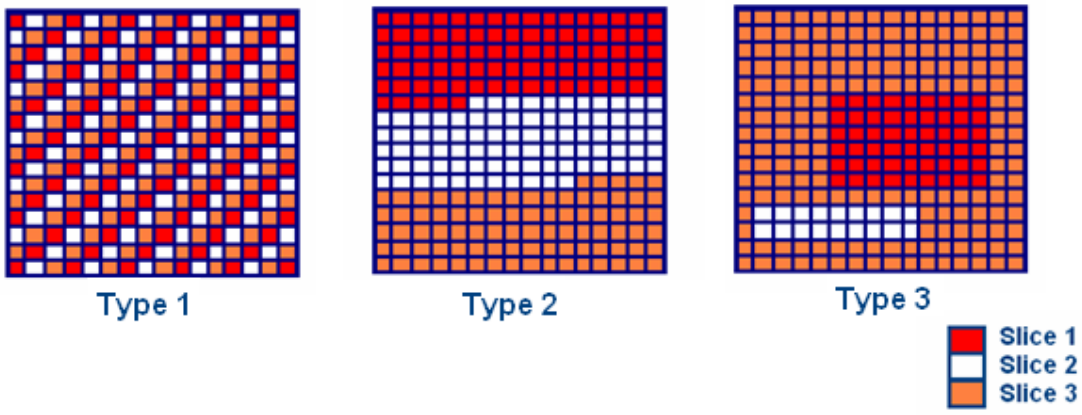


Figure 4.3 Examples of slice's types in H.264.

Each type of slicing represents one frame split in three slices (one of each color). Each one has differents properties:

- Type 1: Scattered slice. The slice group of a macroblock is always different from that of neighboring macroblocks.
- Type 2: Interleaved slice. The slice is defined as the run of consecutive macroblocks in raster scan order.
- Type 3: Rectangular slice structure. Allows coding of a region of interest to improve the coding loss.

Type 3 is allowed by the flexible macroblock ordering (FMO) technique of H.264, that can also allocate to a slice macroblocks not necessarily consecutive in a raster scan order in a picture. FMO defines the slice group as a subset of the macroblocks of a picture and the slice as an integer number of macroblocks ordered contiguously in raster scan order within a particular slice group.

The addresses of the macroblocks are derived from the address of the first macroblock in a slice and the allocation of macroblock to slice group map. The allocation information consists of an array of slice group IDs, one for each coded macroblock, indicating the slice group to which each coded macroblock belongs.

FMO has seven modes, indicated by picture parameter set, each different in how to allocate macroblocks to slices.

4.2.1. Why to use slicing?

No intra-frame prediction takes place across slice boundaries. With that, packet loss probability can be reduced, as the transmission packets are getting smaller (compared to transmitting whole frame as a packet), since the probability of a bit-error hitting a short packet is generally lower than for large packets. Moreover, short packets reduce the amount of lost information and, hence, the error is limited and error concealment methods can be applied in more efficient way.

4.2.2. Drawbacks of slicing

The loss of intra-frame prediction and the increased overhead associated with decreasing slice sizes adversely affects coding performance and requires additional overhead per slice (slice headers). Especially for mobile transmission, where the bandwidth is limited and the packet size clearly affects loss probability a careful selection of the packet size is necessary. H.264/AVC specifies several enhanced error resilience concepts to reduce the artefacts caused by packet losses within one frame.

4.2.3. Evaluation of slicing in H.264

In [7] there is an evaluation of H.264 error resilience tools over RTP/UDP/IP environments. This interesting study considers:

- Oneslice per frame: To compare to the case when slicing is used
- N –slices per frame: N slices (Fixed number of MBs) in one slice group in a picture
- Scattered slice: As defined in FMO.

The methods are evaluated for both, random errors and burst errors.

4.2.3.1. Performance evaluation with random bit errors.

In the simplest case the slice the whole frame is transferred inside one packet. Such packet is considerably longer (especially for I frames) than the packets from N–slice scheme and Scattered slice because this last methods divide the frame in slices and every slice is transferred inside one packet.

In the network the probability of loss of a longer packet is higher than if a shorter one is used. In [7] it is possible to see how 1 slice approach suffers a high PSNR loss of 11.32dB for QCIF resolution.

Scattered-slice leads to the highest PSNR (out of the three presented methods) but needs 23% rate increasing to have this quality. In the study the obtained increment of PSNR is 5.16dB for QCIF resolution. This increment of the bit rate depends on the particular bit rate – in low bit rate, the increment is more perceptible in low bit rate.

Scattered-slice has a rate increasing compared with N-slice because in case of I slice the spatial prediction cannot be made efficiently as we cannot use the top and left neighbours for making the spatial prediction (This neighbours are from other slices, see Figure 4.3), If the spatial prediction cannot be made correctly the slice cannot be compressed like in the case of N-slice.

N slice is able to have a better quality than 1 slice but not like scattered-slice. In the study there are an PSNR increment of 3,25dB for QCIF. The advantage is that in this case the increment of the bit-rate is only around the half of the increment when scattered-slice is used.

The scattered method is able to have better quality because it can use the motion vectors of the neighbouring macroblocks for interpolation of the lost ones, which allows the macroblocks of the lost slice to be concealed.

4.2.3.2. Performance evaluation with burst bit errors

First it is important to see the differences between the burst and the random error environments. With burst errors the errors are concentrated in a few slices so that only these slices will be affected unlike with the case with random errors, where the errors will be dispersed in more packets and slices. So with the same BER in a random error environment will have more packet loss.

In this case the differences between the quality and bit rate of the different methods are smaller than in the random error environment. It is difficult for even a short packet to survive in a burst error, therefore performance difference of different packet sizes can be negligible although the average error performance of a shorter packet is better than the longer one.

4.2.3.3. Evaluation of slice's size

Another thing important to choose about the slices is their size. Choosing a big number of slices (so slices with small size) leads to an increase of the bitrate due to a larger rate consumption by slice header data. But not all is bad, with smaller slices the error resilience will be better so the spatial propagation of the error will be less than if bigger slices were used.

Like showed in [10] the slice header rate increases linearly (middle image), the larger the number of slices.

The upper curve shows the increase of header ratio which is defined as the ratio between header rate and total rate.

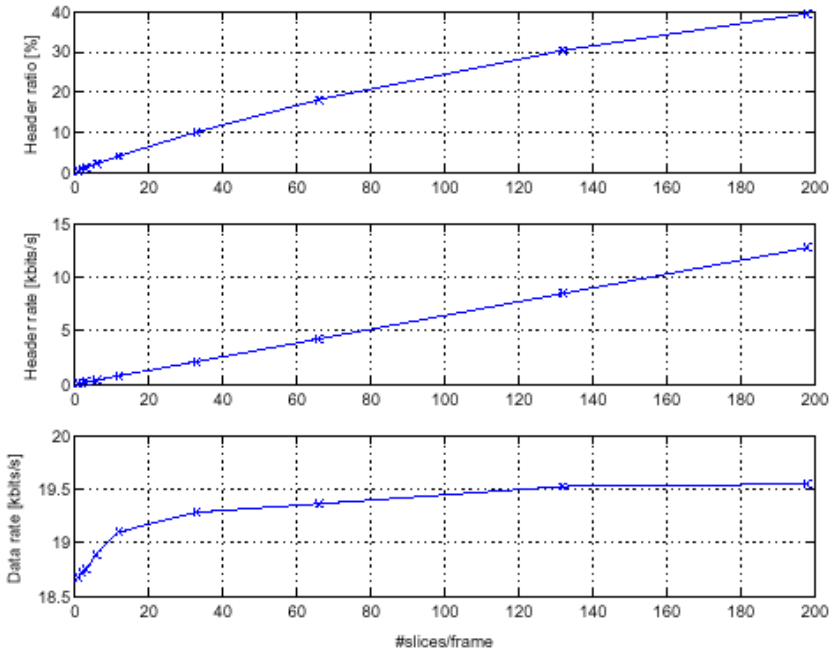
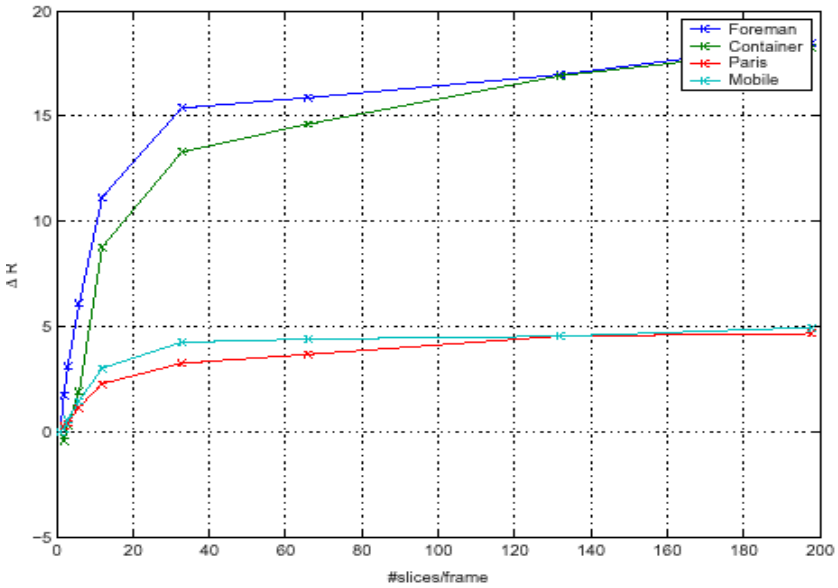


Figure 4.4 Image taken from [10]

Like is possible to see in the right image the rate increases in a strongly non-linear manner with decreasing slice size.



Firgure 4.5 Image taken from [10]

There are so much different between Foreman-Container and Paris-Mobile. The first one need efficient coding of their motion vector which describe quite uniform motion. For Paris-Mobile the rate increase is roughly 5% but for Foreman-Container this increase is about 18%.

4.2.3.4. Conclusions

With slicing error resilience performance increases on average. Besides the scattered-slice have better performance than the N-slice. Also is important to determine the bit rate according to the given network error environment.

For choose a correct number of slices is important to know the video than will be enconded because is the video rely on efficient coding of their motion vectors which describe quite uniform motion the increase of the number of slices will carry a big rate increase.

4.3. Data partitioning and packetization

The main idea is to divide the information in different levels (A,B,C). Each one has different relative importance for the error resilience and protection, and each one has its corresponding robustness.

This option is provided in the extended profile of H.264.

Normally, all symbols of a macroblock are coded together in one single bit string that forms a slice. Data partitioning method creates more than one bit strings (partitions) per slice. Three different partition types are used.

- Header information, including MB types, quantization parameters and motion vectors. This information is the most important, because without it, the other partitions cannot be used. This type is called type A.

- The Intra partition, it carries information about Intra-frames. Requires the availability of the type A partition of a given slice to be useful (type A also contains the prediction type for I frames). In contrast to the Inter information partition, the Intra information can stop further drift and, hence, is more important than the Inter Partition. This type is called type B.

- The Inter Partition, it contains only information about inter-frames, in many cases, this partition is the biggest partition of a coded slice. Inter Partitions are the least important because their information does not resynchronize the encoder and the decoder. In order to be used, they require the availability of the type A partition but not the type B partition. This type is called type C.

Every partition is a NAL unit in different packet. So this method is adding overhead because without it the entire slice goes into the same NAL unit.

The data partitioning is not applied to instantaneous decoding refresh (IDR) pictures, which provide decoder restart anchors and should therefore be heavily protected.

4.3.1. Functionality

The data partitioning is implemented by the source coder because the slice must be adjusted in such a way that the biggest partition does not lead to a packet bigger than the MTU size (usually 1500 bytes).

All data is encapsulated into network abstraction layer units (NALUs), which are put in RTP packet for transmission over an IP network.

Another important thing is that the type B/C prediction residuals are encoded with a context-adaptive variable-length code (CAVLC), which has a simpler and potentially more robust structure than the arithmetic encoding available in the main profile of H.264.

With data partitioning if the Inter or the Intra partitions are missing, the available header information can still be used to improve the efficiency of the error concealment.

4.3.2. Combined methods

Data partitioning for itself is not a powerful system but if the conclusions of the [8] are true, it would be interesting to combine data partitioning with other methods.

The suggested use is that data partitioning will split the information and unequal error protection method will ensure that also the C partition is protected sufficiently.

Maybe it would be interesting to use data partitioning plus synchronization markers. So synchronization markers will be only necessary to protect the C partition because data partitioning protect better the B and A partitions.

Another idea is to use data partitioning with out-of-band method. The problem with out-of-band method is if the out-of-band packet is lost the synchronization marks of the entire slice will be lose. So the out-band-packet can be labeled like one B partition or A depending of the circumstances.

4.4. SP and SI frames.

Previously codecs uses I,P and B frames but H.264 have this three frames types and two new, SP and SI frames.

SP-frames are very similar to P-frames, this frames make use of motion compensated predictive coding to exploit temporal redundancy. The special issue of SP-frames is that SP allows identical frames to be reconstructed even when they are predicted using different reference frames. Due to this property, SP-frames can be used instead of I-frames in such applications as bitstream switching, splicing, random access, fast forward, fast backward, an error resilience/recovery. At the same time, since SP-frames unlike I-frames are utilizing motion-compensated predictive coding, they require significantly fewer bits than I-frames to achieve similar quality.

SI-frames use only spatial prediction as an I-frame and still reconstruct identically the corresponding SP-frame, which uses motion-compensated prediction.

Temporal predictive coding employed in P/B-frames introduces temporal correlation within the coded bitstream. In cases when a reference frame used in an encoder and a reference frame used in a decoder are not identical either due to errors during transport or due to some intentional action on the server side, the reconstructed values of the subsequent frames predicted from such a reference frame are different in the encoder than in the decoder. This mismatch would not only be confined to a single frame but would further propagate in time due to the motion-compensated coding.

In the next points, extracted from [14], it is exposed how these features of SP/SI-frames can be exploited.

4.4.1. Error Resiliency

In the point [Reference to intra-update method point] it was showed that if this method is used the error resiliency is improved considerably. The problem is that intra refresh rate should depend on transport channel conditions, e.g., packet loss and/or a bit error rate. In interactive client/server scenarios, the encoder on the server side decides to encode the slices/macroblocks in the intra mode either based on: the specific feedback received from the client, or the expected network conditions calculated through negotiation, or the measured network conditions. However, when already encoded bitstreams are sent, which is the case in typical streaming applications, the above strategy cannot be applied directly. Either the sequence needs to be encoded with the worst-case expected network conditions or additional error resiliency/recovery mechanisms are required.

SP-frames or slices can be represented as SI-frames/slices that do not use any reference frames. This feature can be exploited in the adaptive intra refresh mechanism discussed above. First, a sequence is encoded with some

predefined ratio of SP-slices. Then during transport, instead of some of the SP-slices their secondary representation, that is SI-slices, is sent. The number of SI-slices that should be sent can be calculated similarly as in the real-time encoding/delivery approach.

4.4.2. Video Redundancy Coding

There are other issues for SP-frames in applications in which they do not act as replacements of I-frames. Video redundancy coding (VRC) is one example of this.

The principle of the VRC method is to divide the sequence of pictures into two or more threads in such a way that all camera pictures are assigned to one of the threads in a round-robin fashion. Each thread is coded independently. In regular intervals, all threads converge into a so-called sync frame. For this sync frame, a new thread series is started. If one of these threads is damaged because of a packet loss, the remaining threads stay intact and can be used to predict the next sync frame. It is possible to continue the decoding of the damaged thread, which leads to slight picture degradation, or to stop its decoding which leads to a drop of the frame rate.

Sync frames are always predicted out of one of the undamaged threads. This means that the number of transmitted I-frames can be kept small, because there Is no need for complete re-synchronization. For the sync frame, more than one representation (P-frame) is sent, each one using a reference frame from a different thread. Due to the usage of P-frames theses representations are not identical. Therefore a mismatch is introduced when some of the representations cannot be decoded and their counterparts are used when decoding the following threads. The use of SP-frames as sync frames eliminates this problem.

4.4.3. Conclusions

This new frame types defined in H.264/AVC can be used to improve the error resilience instead of intra-update. Also to implement VRC the use of SP-frames is high recommended instead of the normal use of P-frames.

SECTION 5. ADDITIONAL RESILIENCE METHODS

5.1. Synchronization marks

5.1.1. The problem

Variable length codes (VLC) are codes having their codewords of variable length. They compact the compressed video bitstream before the transmission. So with VLC code, H.264 (or another codec) can reduce its bit rate. VLC codes are also called entropy codes because the codeword length is chosen according to the probability of the occurrence of that codeword in the stream; to the parts of stream occurring with higher probability shorter codewords are assigned. This results in the highest entropy at the output of such encoder. Highest entropy means that the redundancy of a stream is loss-lessly reduced – compacted. The drawback is that the use of VLCs leads to inevitable error propagation (desynchronization) in case of error prone environment. In the Figure XY the propagation of the error in VLC can be seen.

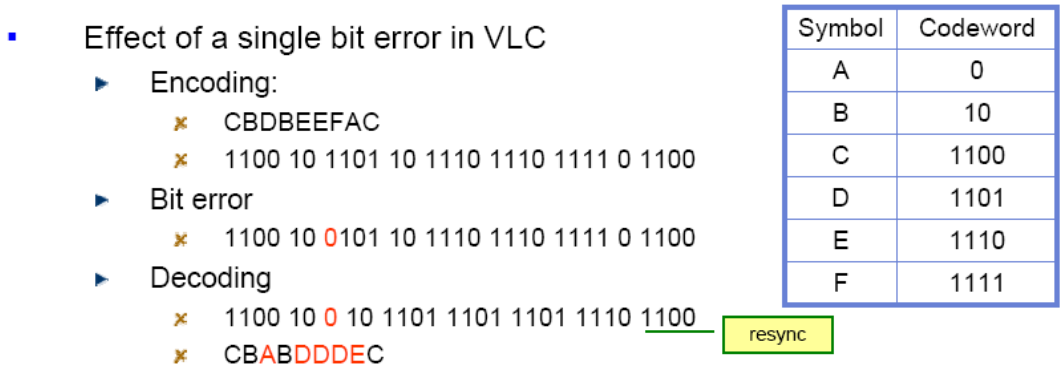


Figure 5.1 Example of VLC desynchronization

It is possible to see that one single bit error will propagate further. How long will be this error? It is not possible to know this in general, it depends on the bit stream and error location. In the Figure 5.1 it is possible to see that the errors have an arbitrary propagation.

5.1.2. Synchronization marks

This is why the synchronization marks are needed. The synchronization marks prevent the error in VLC propagating over the rest of the VLC stream.

So is possible to add special codewords to the VLC table, this codewords are uniquely identifiable within the bitstream. A VLC, a combination of VLCs, or any other codeword combination in the bitstream cannot reproduce the synchronization word. While shorter synchronization words introduce less overhead, they also increase the probability of emulating synchronization words in the bitstream in bit-error-phone environments.

Synchronization marks in video stream do not only provide the bitstream synchronization, they can also ensure spatial synchronization at the decoder. This is achieved by inserting additional fields after the synchronization word for critical information, such as the block address and/or the quantizer value. Moreover, to limit the errors within a slice, data dependencies across slice boundaries within the video frame are usually forbidden by video coding standards. To remove such dependencies, the encoder modifies the coding strategy at the beginning of a new slice. For example, the motion vector of the first block of a new slice is not temporally predicted.

To limit the errors to a small spatial region, synchronization words may be inserted at various locations, either at a uniform spatial interval in the coded frame, or at a non-uniform bit interval in the bitstream.

Using the codewords if there is an error this will only propagate until the next synchronization mark.

In H.264 a start code prefix is standardized for the CAVLC code. A unique sequence of three bytes equal to 0x000001 embedded in the byte stream as a prefix to each NAL unit. The location of a start code prefix can be used by a decoder to identify the beginning of a new NAL unit and the end of a previous NAL unit. Emulation of start code prefixes is prevented within NAL units by the inclusion of emulation prevention bytes.

5.1.3. Placement of synchronization marks

In this part possible ways to use synchronization marks will be discussed to improve the error resilience in H.264.

The standardized start code prefix in H.264 can be used for further improvement of the error resilience and detection. Remember that the synchronization mark is 24 bits (0x000001) long.

In this case the start code prefix will be used to improve the error resilience, for this the decoder will be changed for know that the start code prefix is also use fur this purpose.

5.1.3.1. Where to add the synchronization marks

The start code prefix only separates the NAL units. One NAL unit usually contains one slice. Still, after the first error in the NAL unit we will not be able to resynchronize until the next NAL unit was detected. To localize the error more exact we can add synchronization marks also in different places within the same NAL unit - slice. We have a lot of possibilities:

- Every RLC PDU if UMTS is used, or every number of codewords in a stream
- Every particular number of macroblocks (leads to non uniform placing the marks as different macroblocks have different length after the entropy coding)
- Any other choices

5.1.3.2. RLC packet

For this purpose it will be assumed that the receiver passes the packets with errors to the higher protocol layer like in [9] describing improved error detection for H.264 packet video in UMTS network. If this is not made, does not matter if the error detection is improved because one single error in one of the RLC-packets will cause receiver discarding all the remaining and possibly error-free RLC-packets being the segments of the same IP packet. This is caused by the fact, that CRC is calculated over the UDP/IP packet and therefore the reassembly process will fail if there is an error within the IP packet. As UMTS also adds the CRC to every transport block that is contained within an RLC packet, we are able to know which RLC packets are correct if we allow passing this information together with data to the higher layers. In such case this information could be used for finer than IP-packet based detection – and recovery of the errors.

If the content is entropy encoded video stream, then if one RLC packet is lost all the following RLC packets belonging to the same IP packet will be lost (Even if the receiver passes the packets with errors to the decoder). This is a problem that H.264 has for using CAVLC code. The decoder is not able to resynchronize itself until the next start code prefix.

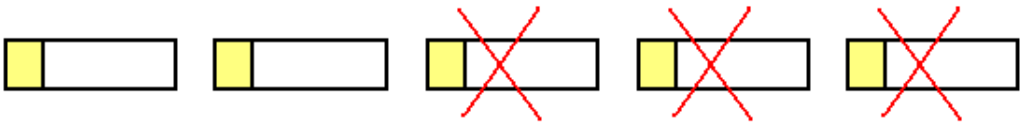


Figure 5.2 Propagation of the error in UMTS

To improve the granularity of the detection it is possible to add a synchronization mark in every RLC packet. Normal length for RLC-PDUs used is 320 bits (for the radio bearers under 384kbps) and 640 bits (for the radio bearers above or equal 384kbps). We will choose 320 bits to study this case, as it is the common case for UMTS multimedia communications.

We have to add the synchronization mark after every L bits

$$L = 320 - Synchmark.lenght$$
$$L = 320 - 24 = 296bits \quad \textbf{(5.1)}$$

We have added an 7,5% of overhead.

These bits will have to be added after the entropy coding cycle.

With this approach the decoder is able to use all the correct RLC packets (i.e. if only the 3rd from 5 RLC packet representing the slice is damaged as shown in the Figure 5.3).

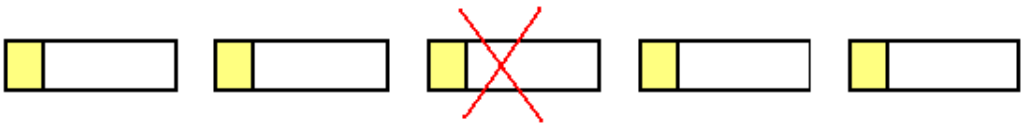


Figure 5.3 Sequence of RLC packets belonging to one single IP packets.

In this case the granularity of the error detection has been improved; we can localize and limit the error to 296 bits for one error in an IP packet.

Now the receiver is able to use the correct information and does not have to discard all following RLC packets.

5.1.3.3. Macroblocks

It is possible to add the synchronization mark after every particular number of macroblocks. It is supposed frames of 9x11 macroblocks codec in one slice of N bytes.

There are 99 macroblocks for each frame, since the slice size is variable the overhead will be studied for different values of it. Normally the slice size will be less than the MTU, the usual MTU (i.e. in Ethernet) is 1500 bytes, The expression to calculate the overhead (increment in size showed in increment of %) is:

$$Ov = \frac{Bytes\_with\_sync}{Bytes\_without\_sync} * 100$$

$$Ov = \frac{(Nbytes\_slice) + (99MBs * 3bytes\_sync\_mark)}{Nbytes\_slice} * 100 \quad \textbf{(5.2)}$$

In Figure 5.4 is showed the graphical representation of this equation. The overhead added to the case when it is added every macroblock is too high.

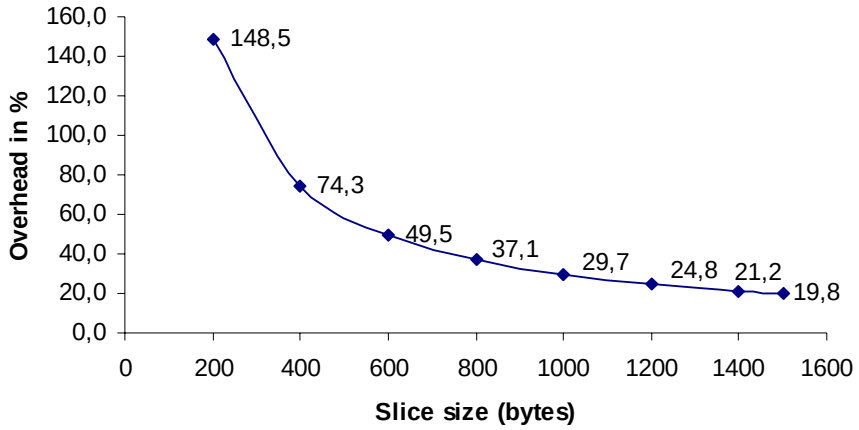


Figure 5.4 Overhead added for different slice's size.

So in Figure 5.5 and Table 5.1 is showed the overhead added for different slices sizes and adding the synchronization mark every a concrete number of MBs.

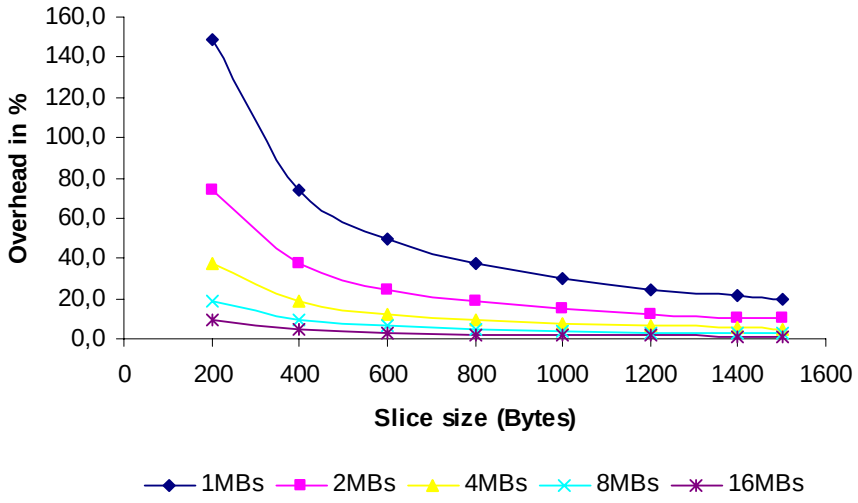


Figure 5.5 Overhead added for different slice's size and every X MBs.

Table 5.1 Overhead by adding sinchronization mark every a concrete number of MBs.(%)

Slice size (bytes)	1MB	2MBs	4MBs	8MBs	16MBs
200	148,5	74,3	37,1	18,6	9,3
400	74,3	37,1	18,6	9,3	4,6
600	49,5	24,8	12,4	6,2	3,1
800	37,1	18,6	9,3	4,6	2,3
1000	29,7	14,9	7,4	3,7	1,9
1200	24,8	12,4	6,2	3,1	1,5
1400	21,2	10,6	5,3	2,7	1,3
1500	19,8	9,9	5,0	2,5	1,2

The blue values in Table 5.1 show the values than are under the overhead of adding synchronization mark in every RLC packet, remember that this was 7,5%.

If this method will be used to UMTS networks all the IP packet will be split into RLC-PDUs in the RNC (Radio Network Controller), so it is necessary to study what happens in this stage.

Let us assume that the synchronization mark is added every 8 macroblocks as can be seen in the Figure 5.6.

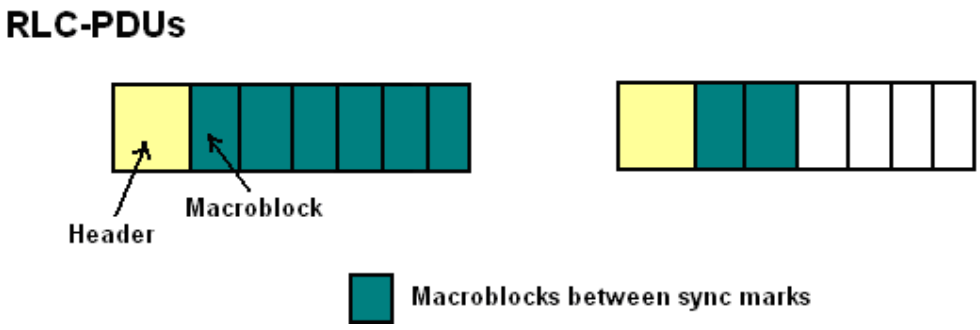


Figure 5.6 Example of synchronization mark every 8 MBs.

If there is an error, this will propagate between two RLC-PDUs, so the CRC of this RLC packets will fail. This is not a big problem if the receiver is able to use the correct macroblocks (in this case the white ones), because if the receiver does not make this the final three macroblocks will be discarded, too. And also the following macroblocks in the RLC packets will experience the same problem, an error will propagate until the start code prefix in the next NAL unit.

Adding the synchronization marks if the decoder is not able to use the correct information inside the RLC packet is not a help.

But if the decoder would be able to do this it would be possible to use the synchronization marks at the macroblock level.

It is difficult to calculate the number of macroblocks that are necessary to put between the synchronization marks. It can be variable depending on the robustness or the bandwidth that has/wants the communication, one could design a feedback mechanism to adapt to the environment or make a first stage of negotiation.

The best possibility seems to be using the synchronization marks in RLC level. It adds the overhead of 7,5% but will improve the granularity of detection. With finer detection it is possible to perform better error concealment of the errors at the receiver (We only will loose one RLC packet, not the whole slice).

5.2. Out-of-band VLC synchronization

Another way to improve the error resilience is adding synchronization out-of-band. It can be useful because it is possible to send all the synchronization of a slice for example in one packet only. It would be possible to have a higher robustness (better protection and/or higher priority) for this packet and ensure that this arrives. With that it is possible to have less errors in the synchronization information resulting in better overall quality at the receiver.

With the synchronization marks in-band it is possible to have also errors in these synchronization marks, so that the information between three synchronization marks will be lost.

The idea is to send a packet with information about the synchronization outside the normal transmission of H.264 stream to try to improve the error resilience with less overhead if possible.

5.2.1. Assumptions

An IP packet of N Bytes is split into several RLC units of 320 bits. This packet contains only one slice.

In the case of the synchronization marks it was studied that a good improvement of the error resilience without adding too much overhead (7,5%) is to add the synchronization marker at the end of every RLC packet. In this case the same approach will be used, the overhead added with this method for adding synchronization every RLC packet will be studied. H.264 uses VLC so with this purpose the out-of-band information will contain the positions of the RLC packet starts within the same IP packet.

Also in this case it is supposed that at the receiver the packet with error will be passed to higher layers of its protocol stack.

The information that will be sent is the position of the beginning of each RLC packet in the payload of the IP packet. With this information the decoder will be able to resynchronize itself in case that an error in a RLC packet occurred. If no error occurred the decoder will not need to use this out-of-band packet.

Normally after the IDR packet the codec sends the NAL units corresponding to the slices. With this method it is needed to send one packet with the synchronization information (the out-of-band packet) before sending each slice. This packet can be also sent 'in parallel' via a different channel, out of the video stream.

5.2.2. Encoding of synchronization marks

It is necessary to calculate how many bits are needed to say to the decoder when every synchronization mark is added. One first approach is to calculate the range of bits that are needed to refer each location.

Let us have an VLC encoded slice of N bytes. There are *M=8N* possible places (1 byte = 8 bits) to add the synchronization mark. How many bits are needed to enable describing the whole range?

$$M = 2^n = 8N \tag{5.3}$$

where *n* is the size of the mark in bits. Therefore we obtain for n:

$$n = \lceil \log_2(8N) \rceil \text{ [bits].} \tag{5.4}$$

Assuming that the number of bits needed for synchronization (*n*) is known, it is necessary to design how many times these bits will be added to the out-of-band packet.

A slice of *N* bytes and RLC packets of 320 bits is assumed, with this it is possible to know how many marks (N_S) are needed.

$$N_S = \left\lceil \frac{8N}{320} \right\rceil = \left\lceil \frac{N}{40} \right\rceil \tag{5.5}$$

The number of bits that will be added will be:

$$L = N_S \cdot \lceil \log_2(8N) \rceil = \frac{N}{40} \lceil \log_2 8 + \log_2 N \rceil = \frac{N}{40} \lceil 3 + \log_2 N \rceil \text{ [bits].} \tag{5.6}$$

Knowing the payload of the out-of-band packet, now it is only necessary to add the header. For not adding more complexity the same protocol stack as in the rest of the transmission will be used (although TCP protocol could provide more reliable transmission, but also trouble with delay in case of retransmissions). This protocols and the length of their header are:

IPv4: 20 bytes
UDP: 8 bytes
RTP: 12 bytes
Total header size: 40 bytes = 320 bits

So the final length of the out-of-band packet (L_f) will be:

$$Lf = Total\_header\_size + L = 320 + L = 320 + \frac{N}{40}\lceil 3 + \log_2 N \rceil \text{ [bits]}. \qquad \textbf{(5.7)}$$

The overhead estimation above was the worst case estimation – we assumed the marks that can address the whole packet area. Another way to encode the marks could be to encode the number of bits belonging to one segment between two synchronization marks. The number of bits needed for this marks depends on the maximum size of the segment, but as soon as the maximum segment is smaller than the whole packet area, we can save the bits needed for the marks.

So another way to calculate this will be possible. To say to the decoder when can be synchronized, the number of bits that there are between the synchronization marks will be transmited. In the next example it will be showed with more details:

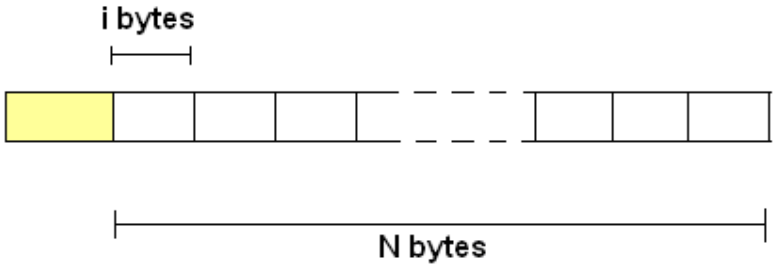


Figure 5.7 Packet to be synchronized.

For example if with out-of-band method it's wanted to resynchronize the decoder every *i* bytes (Figure 5.7), the decoder will only need to know this value (*i*), and not the position of each synchronization mark like in the first approach. So in this case L (number of bits added by out-of-band) will be:

$$L = \lceil \log_2 i \rceil \text{ [bits]} \quad \textbf{(5.8)}$$

If this expression is compared to the expression of L for the first approach it is possible to see that this second is lower than the first one. Only the first term of the first expresion was bigger than this one, N is x times n, where x is the number of synchronizations marks that we will have.

The final lenght of the out-of-band packet (*Lf*) will be:

$$Lf = Total\_header\_size + L = 320 + \lceil \log_2 i \rceil \text{ [bits]} \quad \textbf{(5.9)}$$

The only thing that now is needed is try to find the number of bits (n) to be inserted. Minimum it would expect to detect errors every 40 bytes (RCL packet), for this purposse n will be:

$$M = 2^n \rightarrow n = \lceil \log_2 40 \rceil = 6 bits \quad \textbf{(5.10)}$$

With 6 bits it is possible to define 64 positions, so in the worst case (a packet of 1500 bytes) will be synchronizations every:

$$\frac{1500 bytes}{64 positions} = 23{,}43 bytes \quad \textbf{(5.11)}$$

It is only an example of a value on n, but the variance of the overhead for different values of n will be despreciable because the most part of the overhead added will be from the headers. Now the overhead of this method will be studied.

5.2.3. Overhead analysis

The overhead added is one out-of-band packet per each slice. So the overhead (O_v) will be:

$$O_v = \frac{L_f \cdot 100}{8N} \text{ [%].} \quad \textbf{(5.12)}$$

In Figure 5.8 it is possible to see the overhead that the out-of-band method requires for different sizes for a fixed value of n (n=7).

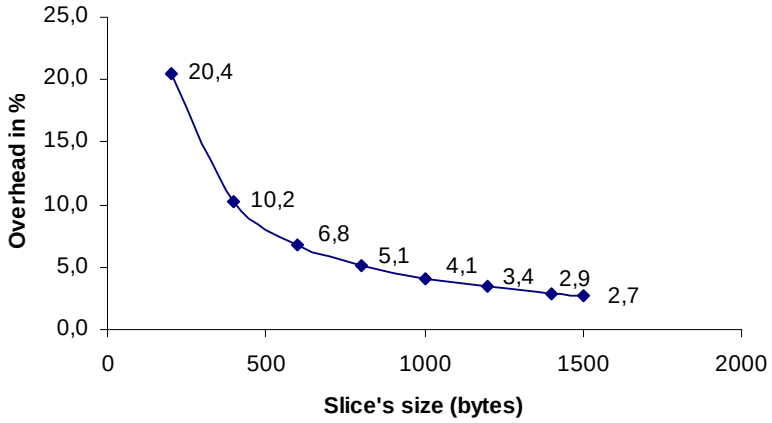


Figure 5.8 Out-of-band overhead for different slice's size.

It is possible to see that the overhead added for slices smaller than 400 bytes is rather high (10,2% to 20,4%) but also this not are the common slice's sizes. For the rest it could be acceptable depending of the environment in witch it has to be used.

The problem of this method is that the most part of the overhead added is from the header. In Figure 5.9 it will see the same figure than in 5.8 but without the header.

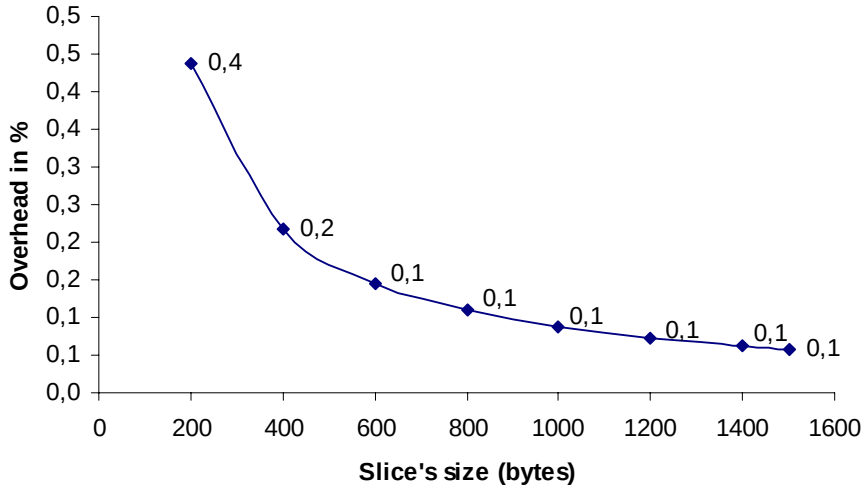


Figure 5.9 Out-of-band overhead for different slice's size without header.

Is possible to see that in this case the overhead added is negligible, but the problem is, how the header can take out?

The solution is in the standard, in the raw byte sequence payloads specification there are 5 reserved bits (reserved_zero_5bits) that may be specified in the future by ITU-T and ISO/IEC. The decoder shall ignore the value of reserved_zero_5bits.

So the idea to take out the header-overhead is:

- Use data partition and try to add the bits for synchronization in the label A for better robustness.
- Use the reserved_zero_5bits to specify the length of the information that will be added.
- Tell to the decoder that have to take the value of reserved_zero_5bits and not to ignore the value.

The decoder will take the packets labeled as A, and will read the reserved_zero_5bits (*b*) to know the length of the information about synchronization included in the packet. The decoder will decode the packet type A normally and when it finishes the normal decoding process it will know that there are *b* new bits that specify where can be resynchronized. In this case we are using a packet that will be sent anyhow, so this method is not adding the header information and the overhead can be calculated without it. The overhead (*Ov*) will be:

$$Ov = \frac{(reserved\_zero\_5bits + nbits)}{8Nbits\_per\_slice} * 100 \qquad \textbf{(5.13)}$$

The graph of the overhead for this case is (n=7):

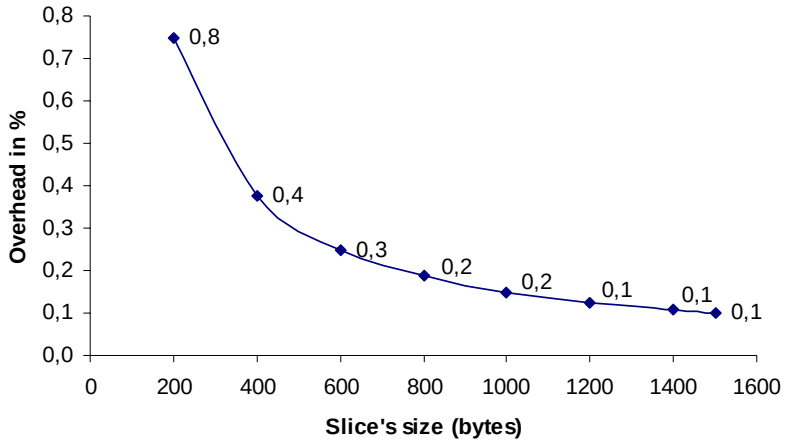


Figure 5.10 Out-of-band overhead using reserved bits.

As it is possible to see in the Figure 5.10 the overhead added is negligible.

5.2.4. Comparison of in-band and out-of-band signalling

It is necessary to remark that out-of-band option is very interesting because the encoder is able to change the frequency of the signaling only to change the bits that are added inside the A packet. So it is possible to change the robustness of this method depending on the conditions of the communication environment. Another good point is that de old decoders will function properly because this method uses reserved bits that normally the decoders do not take care about.

Only the problem is that if the A packet is lost, it will be lost all the synchronization of the slice. It was said before that this packet have better protection than normal ones (unequal error protection can be applied), but if in one case it is lost the resulting error will be greater than if the in-band method is used. But on the other hand, in the case of in-band signaling, errors in synchronization marks cause the same problems. It is obvious that out-of-band solution only makes sense if we have a chance to use uncorrelated channels for transmitting the positions and video stream, or if we have a possibility of unequal error protection and priority handling in the underlying network.

5.3. Parity bits

Parity bits are used as a form of forward error control. They are inserted at the sender and after the transmission further checked by the receiver to find out if errors occurred. According to the type of code and amount of redundancy it may be possible to localize and correct the errors. Simplest example to detect the odd number of errors is the single parity check code: before sending the bitstream the encoder adds one bit of parity corresponding to the sum modulo 2 (XOR) over the bits in the stream. The decoder calculates the parity of the received bitstream and compares it with the last bit. If this comparison doesn't match the decoder has detected an error. For our example with single parity check code it is obvious that it is not possible to detect even number of errors and that it is not possible to find out how many errors occurred. To do so, more than one bit redundancy would be needed.

There is no standardized method to use parity bits in the bitstream of H.264 to increase the error resilience and enable better localization of the error.

5.3.1. Main functionality

This method is only for detection of errors. The encoder will be able to know where there are the errors to make the error concealment later. If an error occurs it will not be necessary to discard all the information between the synchronization marks because the decoder will be able to know where the error have occurred and does not loose the synchronization.

With VLC codes if an error happens, the decoder looses the synchronization but if the decoder knows the location of the error, the decoder will be able to discard this erroneous fragment and continue decoding the rest of the transmission.

The main idea is to add one parity bit every N bits for checking in the decoder if these bits had arrived properly. With other methods (Synchronization marks, out-of-band) it was possible to ensure the bits between every 320 bits approximately.

The decoder will receive the bitstream encoded with the VLC code. It will take the bitstream corresponding to the first macroblock for calculating the parity of this and compare the calculated parity with the next bits corresponding to the parity calculated by the encoder. The encoder will be able to detect the error and continue decoding the next macroblock.

Further we will assume the QCIF resolution (144x176 pixel) resulting in 9x11macroblocks per frame. It is considered that the frame is encoded in only one slice of N bytes.

5.3.2. Study of Overhead

The overhead will be calculated for the case that one bit of parity is added in every macroblock. Firstly it's calculated the bits added (*L*) for the parity of the entire slice.

$$L = number\_of\_MBs\_per\_frame = 11 \cdot 9 = 99 \text{ [bits]}. \qquad \textbf{(5.14)}$$

The overhead (Ov) added will be:

$$Ov = \frac{100L}{8N} = \frac{´9900}{8N} = \frac{1237.5}{N} \text{ [\%]}. \qquad \textbf{(5.15)}$$

In the Figure 5.11 it is possible to see the overhead (%) added for different slice sizes

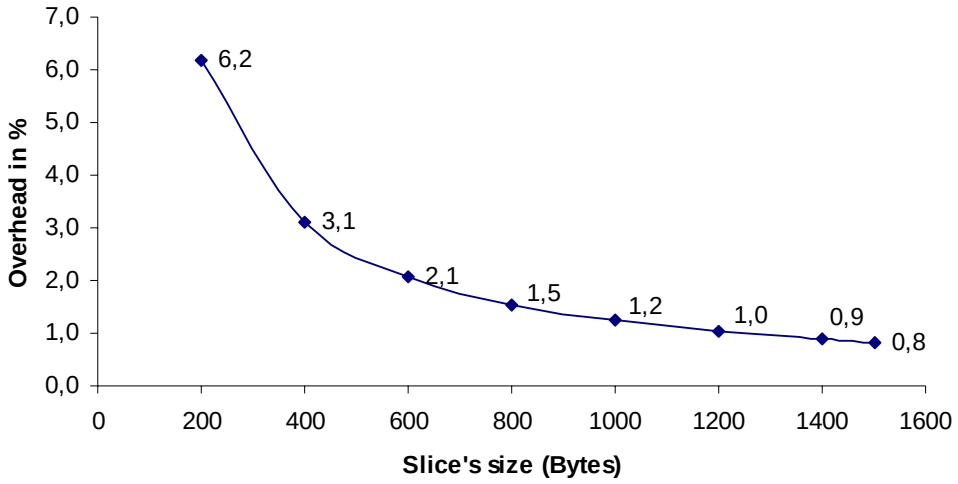


Figure 5.11 Parity bits overhead for different slice's size.

5.3.3. Study of Efficiency

In similar case with the synchronization marks and out-of-band method it was possible to ensure the information from every 320 bits with about 7,5% of overhead. In this case with parity bits the information is ensured in average every N_v bits

$$N_v = \frac{Number\_of\_bits\_per\_slice}{Number\_of\_parity\_bits\_per\_slice} = \frac{8N}{99} \text{[bits]}. \qquad \textbf{(5.16)}$$

In the Figure 5.12 it is possible to see the granularity of the method for different number of slice sizes.

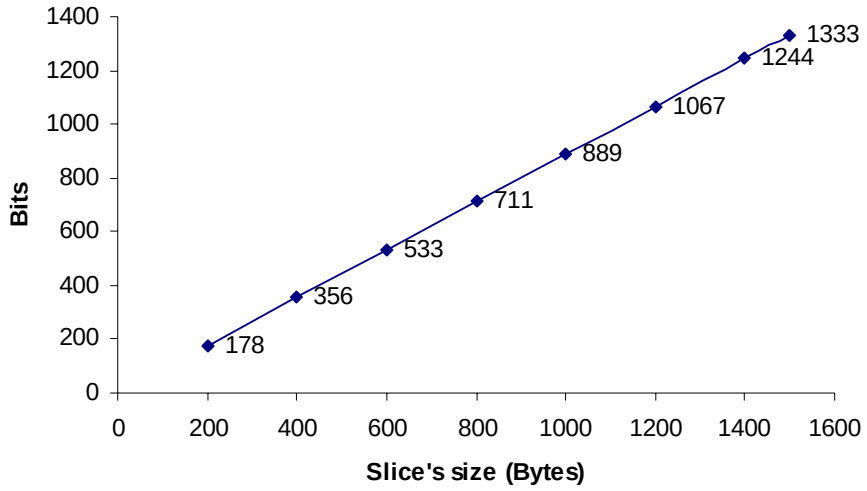


Figure 5.12 Granularity in Bits of the method

But this method is only efficient when there are only even errors. Because, if there are a pair number of errors inside one macroblock the decoder will calculate the parity, and will label this like correct. To add robustness to this method is possible to add more bits for each macroblock in the Figure 5.13 it is possible to compare the overhead of this method for different number of parity bits.

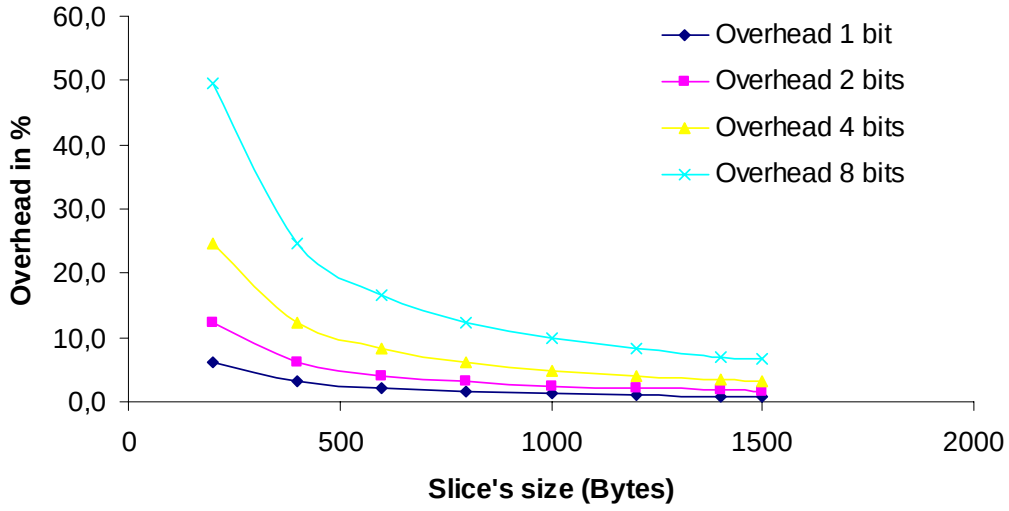


Figure 5.13 Parity bits overhead for different size of the parity mark.

Table 5.2 Parity bits overhead for different size of the parity mark.

Slice size (Bytes)	Overhead 1 bit(%)	Overhead 2 bits(%)	Overhead 4 bits(%)	Overhead 8 bits(%)
200	6,2	12,4	24,8	49,5
400	3,1	6,2	12,4	24,8
600	2,1	4,1	8,3	16,5
800	1,5	3,1	6,2	12,4
1000	1,2	2,5	5,0	9,9
1200	1,0	2,1	4,1	8,3
1400	0,9	1,8	3,5	7,1
1500	0,8	1,7	3,3	6,6

Like showed in the Table 5.2 even with adding 4 parity bits the overhead added is similar that with the synchronization and the out-of-band method.

Depending of the overhead's tolerance of the channel the correct number of bits will be choosed.

The complexity added is similar to the one of the synchronization method because the decoder in this case only have to calculate the parity (AND operation) and in the synchronization method it has to compare the received bitstream with the synchronization marker. These two operations have approximately the same process time.

5.3.4. Conclusions

Another problem is that this method will work in wireless medium. In this medium normally there are no discrete errors; the most common errors are bursts. This method is not designed properly to be capable to resist this type of errors.

Also there are the problem if the error is in the parity bits the most probably thing is that the decoder will take an erroneous decision, because the decoder can confuse part of the parity bits with one VLC value.

In all this cases of errors the result will be the same the decoder will be desynchronized until the next start code prefix in next NAL unit. When synchronization markers or out-of-band are used if an error occurred is only lost the information between the synchronization points (approximately every 320 bits).

This method can detect the errors with better granularity than the other exposed when there are a lower number of errors. But have the problem that if there is a high percentage of errors the granularity of this method decreased. In this case the error is the same that if no parity bits were added.

So this method is only recommendable in environments with low percentage of errors or to be used with another method, parity can increase the robustness of the transmission with no overhead a only a little overhead

5.4. Watermarking

This method started at 1979, but it was only in 1990 that it gained a large international interest. Watermarking is a technique to add imperceptible information into multimedia data through its slight modification.

A watermark is a digital code embedded into a video sequence. The basics of watermarking, according to [13], say that a watermark to be effective should have the following properties:

- Unobtrusive: The watermark should be perceptually invisible.
- Unambiguous: Retrieval of the watermark should uniquely identify the copyright holder of the video sequence if used for property reasons.
- Robust: The watermark should either be impossible to remove or removing it should result in severe video degradation.

A watermark should be robust to the following attacks:

- Common signal processing: The watermark should still be retrievable even if common signal processing operations, such as resampling, requantization, digital-to-analog and analog-to-digital conversions and common signal enhancements to image contrast or color.
- Common geometric distortions: The watermark should survive geometric operations such as rotation, translation, cropping and scaling.
- Collusion and forging: The watermark should be robust to collusion by multiple purchasers who each own a different watermarked copy of the video sequence. It should also be impossible for colluders to combine their video sequences to generate a different watermark with the intention of framing a third party.

5.4.1. Watermarking and predictive coding

If inserting a watermark changes some value that is used to predict some other value, the distortions generated by inserting the watermark will spread to the predicted value.

If there is a whole sequence of predictive coded coefficients, each based on the previous one, and the watermark modifies all of them, the distortions will accumulate and the last coefficient will get all distortions, destroying it. This is obviously not unobtrusive.

In [13] three methods are proposed to solve this problem:

- For each coefficient that is changed, determine all values that must be changed too to compensate for the changed prediction. This would completely solve the problem, but would make the insertion algorithm much more complicated, since coefficients are generally predicted from coefficients of previous frames, which means that there will be many values to keep track of.
- Do not insert watermark into coefficients that will be used to predict future coefficients. Unfortunately, a vast majority of all coefficients will be used to predict coming coefficients, and those that are not can generally be completely removed without degrading the quality, which would make the watermark easily defeated.
- Do not insert the watermark into coefficients that use predictive coding themselves. This will not solve the problem completely, since a coefficient can be used to predict other coefficients even if it does not use predictive coding itself. However, at least the error generated will be constant rather than accumulating.

5.4.2. Use of watermarking

Normally watermarking is used for copyright protection, labelling, monitoring, tamper proofing, conditional access, etc. The most frequent is to use watermark to identify the owner of the video sequence, this is also know as fingerprinting.

In this case it will be used to add a mark in each slice, MBs or group of MBs, for then, in the encoder try to find this mark. If the mark is found the part to be decoded is correct, if not, the part will be labelled as erroneous.

5.4.3. Adding the watermark

Since H.264 uses transform coding, and its video sequences are stored in the transform domain, the watermark should be easily inserted into the transform coefficients. This means that a watermark w can be described as a sequence *w1* to *wN*, where *N* is the number of transform coefficients in the video sequence.

5.4.4. Conclusions

This special method has the advantage that no overhead is added, only some changes are needed in the transform coefficients; for then, in the decoder detect the watermark to see if the reception is ok.

The problem is the complexity, where add the watermark for don't have a loss of the image quality? And can be this operation made in real time or this will need a preprocessing before the sequence will be send?

SECTION 6. PROPOSED METHOD

A lot of methods had been studied to improve the error detection in H.264. In this last section a combined method will be proposed. There is no one single solution to the problem, we will propose one possible approach but other combinations are possible.

First is needed to configure correctly H.264 (Slicing, Intra-update, etc.) using the concepts showed in the 4th section. Then one combined method between the ones in the 5th section will be proposed. With this two decisions the error detection will be improve in H.264 without adding too much overhead.

The method proposed will be a combination of out-of-band signalling and parity methods. These methods will be combined using data partitioning to minimize the overhead and maximizing the robustness.

Out-of-band method had showed less overhead than the other methods using one enhancement in data partition A. Using the 5 reserved bits this combination is able to resynchronize de decoder without to much complexity and overhead. All this is explained in 5.2.3. This combined method will resynchronize de decoder in case of error detection.

Another interesting method was the one using parity bits, parity bits are very useful when there are not so much errors. If in a MB there is one error the decoder is able to know that there is one error in this place and can discard this information. In this case no resynchronization is needed and if the decoder doesn't use this information then no temporal propagation of the error will take place.

All the decisions depend on the overhead that can be added and the robustness necessary.

Now we will take decisions about all the necessary parameters.

6.1. Slicing mode

Like showed in the section 4.2 of this document scattered-slices have better performance than others slices types. The only problem is the increment in bit rate necessary. In mobile networks the bitrate is very restrictive so for this environment slices of the same number of bytes will be a correct choice. In the section 4.2 it is possible to see how this type of slices have also a good performance with less bit rate than others. This slices have the same properties of N-slices but with the decoder will send always packet with the same length, so the sequence will be split in a very efficient way in the RNC.

6.2. Intra-update

For this a correct frequency of I-MBs will be chosen. Depending on the tolerance of overhead in the environment we will take a different number of I-MBs per slice. For this it's only necessary to see the Figure 4.2 and take the correct value.

Another important thing is to decide how this I-MBs will be inserted. The interesting thing would be to add this in the regions with more motion; the problem is that this adds more complexity. This is not a good thing in mobile networks. For this reason randomly chosen will be taken.

6.3. Data partitioning and out-of-band

This will be used to ensure the important information. In this case some extra information will be in DPA. In this label there is the most important information, and in this case also we will add information of synchronization. The method proposed in 5.2.3. will be used to add extra information in the raw byte of the DPA packets.

With this the granularity of the detection will improve because the problem of VLC desynchronization is solved. Also one interesting thing is the backward compatibility; the old decoders will not have problems to decode this new enhancement, they only will discard this extra information.

6.4. Parity bits

With out-of-band we will improve the granularity but also we will lose information in case of errors. We can make mistakes in the detection and this will be used to prediction of other frames. With parity bits we add extra bit in every macroblock to detect possible errors, if there are one error the decoder can discard this macroblock and don't use this to prediction of other ones.

6.5. Experiments

We will show what happen if our method is not used for minimize the error propagation. We will make this in two ways: with concealment and without it. We are using a QCIF resolution foreman sequence adding I frames every 10 frames. The error is inserted in one MB in an Intra frame.

6.5.1. Without improvement method

Without our additional detection method the whole slice is lost, in this case the propagation of the error is very high, in the Figure 6.1 it is possible to see this. No error concealment is used, the green areas are MBs with errors.

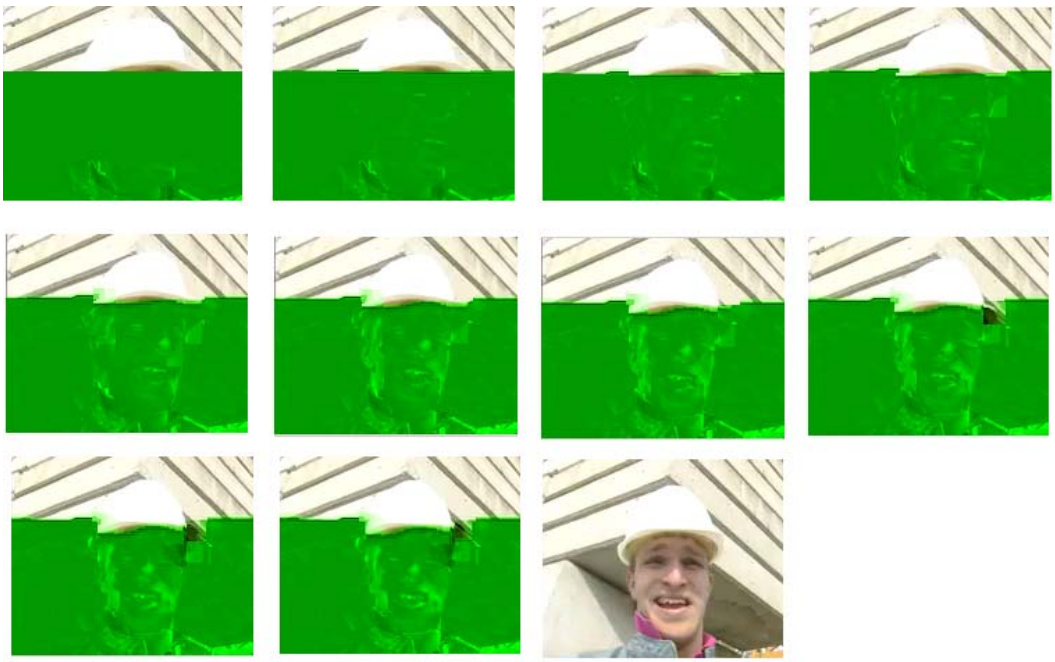


Figure 6.1 Sequence decode without concealment

There is a very high spatial and temporal propagation of the error until the next I frame when the encoder is resynchronized. Now we will decode the same sequence with temporal error concealment [15] (motion compensated prediction for P frames and boundary matching for I frames) to try to minimize propagation of the errors.

Figure 6.2 Error concealment without improvement method

Now with the error concealment the impact of the errors is lower. But also now it is possible to see errors in the regions with more motion, the region is too big for conceal it correctly.

6.5.2. With improvement method

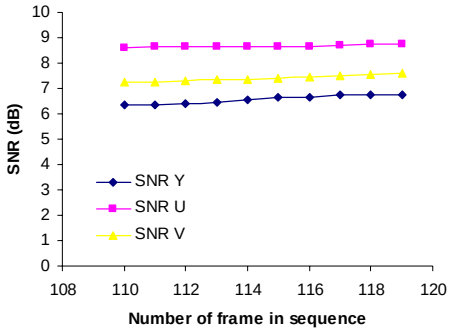
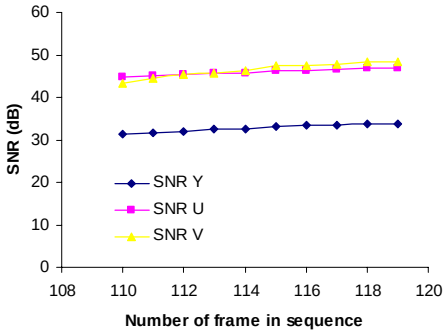
Now the improved method will be used for error detection, so we will make this with and without concealment. In this case the region with errors will be smaller because the decoder can resynchronize and discard errors very quickly.

Figure 6.3 Improvement method without error concealment

The decoder had been resynchronized very quickly but we see that the error is propagated first spatial and then temporal. Using concealment this will be better. In Figure 6.4 is showed this.

Figure 6.4 Improvement method and error resilience

In this case the errors are negligible. Only one MBs really was erroneous so the decoder was able to make concealment and stop the spatial propagation of the error. In the Figure 6.5 it is possible to see comparative SNRs graphs for the 4 cases.

	Without Concealment	With Concealment
No method		

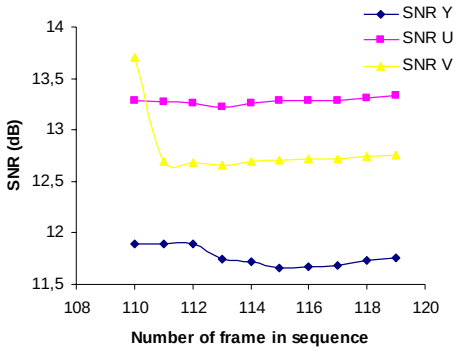
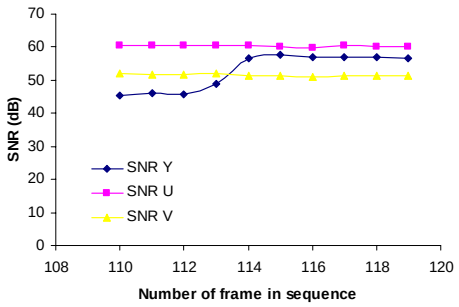
With method		

Figure 6.5 Comparative

It is possible to see that using concealment and our error detection the SNR increases approximately three times than without it. The method is good for stopping the error propagation, following all the steps explained in the proposed method it is possible to achieve good results.

6.6. Conclusions

Using all this parameters it is possible to improve the error resilience, robustness and granularity of H.264. But it is necessary to be careful choosing the parameters. First it is necessary to study the environment in which the video will be sent to choose the correct settings. This can be done also in an adaptive manner by means of a feedback mechanism.

Other combinations are possible but this is the best if a low complexity is expected. This is an important thing because in mobiles is important this thing. Also a good thing is that the changes needed to the codec are only a few.

Backward compatibility is also a good point, except parity bits the other are totally compatible with the old decoders. But add parity compatibility is not so much difficult so this is an unimportant problem.

ABBREVIATIONS

AVC	Advanced Video Coding
B frame	Bi-predictive frame
CAVLC	Context Adaptive VLC
CAVLC	Context Adaptive Variable Length Code
CBR	Constant Bit rate
CN	Core Network
CRC	Cyclic Redundancy Check
FMO	Flexible Macroblock Ordering
I frame	Intra-predicted frame
IDR	Instantaneous Decoder Refresh
IP	Internet Protocol
IPv4	IP version 4
MB	Macro Block
MTU	Maximum Transport Unit
NAL	Network Adaption Layer
PDU	Protocol Data Unit
P frame	Predictive frame
PSNR	Peak (Pixel) Signal to Noise Ratio
RLC	Radio Link Control
RNC	Radio Network Control
RTP	Real Time Protocol
SNR	Signal to Noise Ratio
SS	Streaming Server
TCP	Trasmission Control Protocol
UDP	User Datagram Protocol
UMTS	Universal Mobile Telecommunications System
VCL	Video Coding Layer
VLC	Variable Length Code
VRC	Video Redundancy Coding

BIBLIOGRAPHY

[1] Wiegand T., Sulliven G., Bjontegaard G., Luthra A., "Overview of the H.264/AVC Video Coding Standard", *ieee transaction on circuits and systems for video technology, vol 13, nº7,* (2003).

[2] Richardson I.,"Prediction of Intra Macroblocks", www.vcodex.com/h264.html (2002).

[3] Richardson I.,"Prediction of Inter Macroblocks in P-slices.", www.vcodex.com/h264.html (2002).

[4] Richardson I.,"Transform and quantization.", www.vcodex.com/h264.html (2002).

[5] Stockhammer T., Hannuksela M., Wiegand T., "H.264/AVC in Wireless Environments", *ieee transaction on circuits and systems for video technology, vol 13, nº7,* (2003).

[6] Richardson I.,"H.264 Variable Length Coding", www.vcodex.com/h264.html (2002).

[7] Jung B., Hwang Y., Jeon B., Kim M.,Choi S., "Error Resilient Performance Evaluation of MPEG-4 and H.264", School of Information & Communication Engineering (korea) and Electronics and Telecommunications Research Institute (korea).

[8] Al Moghrabi A., Nemethova O., "Error detection for H.264 packet Video in UMTS Network", TU Wien, (2004).

[9] N D. Dao. W.A.C. Fernando, "Channel Coding for H.264 Video in Constant Bit Rate Transmission Context over 3G Mobile Systems", Asian Institute of Technology, Thailand.

[10] Hallback T., Olsen S., "Error Robustness Evaluation of H.264/MPEG-4 AVC", Department of Telecommunications, Norwegian University of Science and Technology, Norway.

[11] Coté G., Kossentini F., "Optimal Intra Coding of Macroblocks for Robust H.263 Video Communication over the Internet", Department of Electrical and Computer Engineering, University of British Columbia, Canada (1998).

[12] Turletti T., Huitema C., "Videoconferencing on the Internet", *IEEE/ACM Transactions on networking, vol 4, nº3,* (1996).

[13] Bergkvist D., "Implementation of a Watermarking Algorithm for H.264 Video Sequences", Institutionen för systemteknik, Linköping Universitet (2004).

[14] Karczewicz M., Kurceren R., "The SP- and SI-Frames Design for H.264/AVC", *IEEE transaction on circuits and systems for video technology, vol 13, nº7,* (2003).

[15] Sun M., Reibman A.R., "Compressed Video over Networks", Marcel Dekker, Inc., New York, (2001).