Master's Thesis

Real-time monitoring and prediction of Internet Video

By

Jordi Rey Morales

Department of Electrical and Information Technology
Faculty of Engineering, LTH, Lund University
SE-221 00 Lund, Sweden

Abstract

Video Internet services are increasing every year. High-Quality TV over Internet is a service offered by many companies in many countries, and video streaming websites are within the most popular sites on the Internet. Because of that, service providers have a major concern in giving to their customers the best quality in their services. In order to guarantee the quality on these services, a Quality of Experience monitoring tool is required. The main goal of this thesis is to develop a tool to monitor the Quality of Experience of video over Internet services.

In order to be functional, this monitoring tool has to be usable in a real scenario. In this case, a real scenario implies that this tool is used in strategic points of the network and the only available means is the data stream that goes through the network. The most important feature of this tool is that it has to be functional in real-time. It has to be so because its goal is to detect problems before these affect the quality of the video offered to the customers and avoid their complaints.

In this thesis a lightweight method to monitor the Quality of Experiences is implemented. It is designed to predict in a fast way the quality of the video using only the data stream: information in the Internet protocols headers and in the coded video data. In order to achieve this, the structure of the packets has been studied, including the format of the protocols headers and the video codec encoding performance. Then the software has been designed to receive the stream and calculate the parameters that give an estimation of the Quality of experience.

This method has been tested to validate its performance and conclude whether it is or not a valid tool to be used in a real scenario or not.

Acknowledgments

This Master’s thesis would not exist without the support and guidance of my supervisor Maria Kihl, to whom I am grateful for the opportunity to study under her supervision she gave me. I would also like to thank Iñigo Sedano for providing me background information and contents for my researck. At last, I also want to thanks Pere Parés for his advice in drafting this report.

Jordi Rey Morales

Terminology and abbreviations

IPTV	Television over IP
P2P	Pear to Pear
RTP	Real-time Transport Protocol
RTCP	Real-time Control Protocol
UDP	User Datagram Protocol
TCP	Transmission Control Protocol
Port	Abstraction that transport protocols use to distinguish among multiple destinations within a given host computer.
RTP packet	A fixed RTP header and the payload data
RTP payload	The data transported in a RTP packet
RTP profile	Default static mapping of payload type codes to payload formats.
RTP session	Association among a set of participants communicating with RTP
RTCP packet	A fixed header followed by control data that vary depending on the type of packet
SSRC	Synchronization Source
CSRC	Contributing Source
QoE	Quality of Experience
QoS	Quality of Service
DCT	Discret Cosinus Transformation
IDR frame	Intra-coded frames. Reference frames for video reconstruction.
P frames	Predicted frames. These frames need previous frames to be decoded to be reconstructed.
B frames	Bidirectional predicted frames. These frames require previous and future frames to be decoded.
GOP	Group Of Picture. Interval between IDR frames
VCL	Video Coding Layer
NAL	Network Abstraction Layer
PSNR	Peak Signal to Noise Ratio
SSIM	Structural SIMilarity
MS-SSIM	Multi-scale SSIM
VSNR	Visual Signal to Noise Ratio
VQM	Video Quality Metric

Table of Contents

CHAPTER 1

1 Introduction

1.1 Background.

In this project we want to develop a software to monitor the quality of the services that offer video content through Internet. Many agents interact to make it possible, but to acquire a basic idea of the context in which the project is situated we have to know about Internet, video streaming services and the technology used.

1.1.1 Internet

Internet has become a universal tool, with around 2.000 millions of users all over the world. It is a gateway to all kind of knowledge and information.
Its usage evolves driven by many agents. As new services appear, new fashions and tendencies can guide the main consumers to change their behavior. Political and commercial agents can modify the tendencies of the Internet usage as well.
Internet main uses are for communication, social networking and media sharing. Although, it is more important to take a look at the traffic dedicated to each service, as i.e. media sharing websites (i.e. mediafire.com), P2P services (i.e. BitTorrent) have a bigger load of traffic than social services like twitter or mail. Only Facebook traffic can be compared to them, as it is much more popular than other social sites. Another important service that uses the Internet bandwidth is High Quality Video over the Internet, which also implies a huge load of traffic.

1.1.2 High Quality Video over Internet

There are basically two services of high quality video delivering over Internet: video streaming websites and TV over Internet services.

The most famous video streaming websites are http://www.YouTube.com/, http://vimeo.com/, http://www.metacafe.com/ , http://www.hulu.com/ (only

USA) and http://www.veoh.com/ [1]. Each of them has their own differences, but all of them are sites where users or companies upload videos for the public to stream and watch them.

Just to make ourselves an idea of the load of traffic that video streaming websites generate, we can take a look at YouTube, the most important of them: every day more than 4.000 millions of videos are played from YouTube website [2]. With an average length of 4’12 minutes, it means over 280 millions of hours of video streaming every day. The most popular video categories are music and entertainment and fun, including videos with over 700 millions of views [3].
These statistics make it easy to understand that video streaming sites add a huge load of traffic to the broadband network.

On the other hand we’ve got the TV over Internet services: IPTV and Internet TV. IPTV delivers the conventional television contents over a broadband connection. In Sweden there are already some broadband Internet providers than offer this service, such as Telia and Bredbandsbolaget. Internet TV is a service provided by the content providers, such as television channels or producing companies. Internet TV consists on the delivering of the contents from the company website, like video streaming websites but with exclusive content from the company. Usually these websites offer both the content they offer in the television in real-time and old programs to be seen over again.

IPTV services are in a constant growth: nowadays there are five times more subscribers than in 2007 [4]. In 2008 there were already 685 companies worldwide deploying IPTV services [5] and working towards offering a more complete service in the TV.
One of the main differences with the digital or satellite television is that instead of receiving all the channels and choosing one on the decoder of the TV, the user chooses which channel or service wants to receive and this is the only information that is received in the costumer end. With that, the resources required are much less and there is margin to increase the quality of the videos and the television. A standard quality IPTV channel requires a connection of 1.5Mbps while a high definition one requires 8Mbps.

Both videos streaming websites and TV over Internet services need a special monitoring of the network distortion on the information transmitted. They need it in order to provide a good QoE to the customer. The reason they need special attention in this aspect is because the degradation introduced by the network is reflected in the video decreasing its quality. This situation cannot be tolerated because customers will complain and even cancel the contract in the case of the IPTV or not use the website anymore in the video streaming sites case.

1.1.3 RTP & H.264: new streaming technologies

In order to provide higher quality to the Internet video services, the research in the Internet communication area is focusing not only in giving larger bandwidths but also in optimizing the bandwidth available.

Basically, the improvements can be done by developing new Internet protocols and new video codecs.

The research in finding new Internet protocols works towards achieving higher bit rates in the available bandwidth. That can be done, for example, by trying to invent protocols with smaller headers so there’s more space in the packets for the video data. The difficulty lies on the way to do it without decreasing the reliability of the network.

For that purpose, RTP has been developed. RTP works in the application layer, over UDP. The UDP protocol has an advantage in front of the TCP: it’s much faster. There is a reason for this: UDP has no form of flow control or error correction. This fact makes UDP headers much shorter than TCP and permits much more data to fit in every packet. If reliability is wanted to be guaranteed in the UDP connection, the layer above (application layer) is the one which has to take care of it. In this case, RTP can be useful.

As we said before, developing new video codecs is also a right way to optimize the streaming speed. In this case, the optimization consists on compressing the video and in consequence making the size of the stream smaller. In this aspect, nowadays, H.264 is the best video codec. It has a great advantage over MPEG-4: the H.264 bit rate can be 50% less than the MPEG-2 bit rate for the same video. If we put it in another way, with the same bit rate it delivers much higher definition to the video coded.

1.2 Problem description and goals

Video streaming services are among the most influenced by the network distortion due to it has a direct impact on the quality of the video. Therefore, video services requires Quality of Experience monitoring and prediction in order to guarantee good definition in the video. The most important is that this monitoring needs to be done in real-time, so it will be able to detect troubles and avoid clients' complaints. The main goal of this project is to develop software able to monitor the QoE of the Video over Internet service.

A difficulty of this project is that QoE is a subjective metric [6] as it relates on how the viewer perceives the video. This kind of metrics is usually hard to quantify, so a solution is to estimate it as accurately as we can by using objective metrics are usable.

In a real scenario, the monitoring should be performed inside the network, preferably near the client-end as that is where the network distortion can be appreciated. In these situations, the only means available to monitor is the distorted stream, not the original video file. In order to place the problem in this real scenario, in this project a lightweight software that only uses the video stream will be developed in order to monitor the QoE in real-time.

Once the software is developed, it will be tested in a network demo system in order to check the correct performance of the software and the method it is based on.

1.3 Limitations

There are some limitations that are not considered in this project that could be considered to position it in a more realistic situation. Also, the project is focused in a specific situation based on the Internet protocols and video codec used.

About the first limitation mentioned, in this project only packet loss is considered from all distortions introduced by the network. In a real scenario, the stream would also be affected by packet duplication, corruption and reordering or jitter.

Other situations could be analyzed in terms of Internet protocols and video codecs. In this project the streaming of H.264 coded video over RTP is studied, because they are the supposed to be the fastest protocol and codec, but nowadays it’s not widespread. Even if every day more videos are coded using H.264, most of the video streaming websites still use TCP instead of UDP/RTP.

The last limitation is the full-reference metric the prediction method is based on. It was chosen in previous work between several metrics because of the very good performance, but other full-reference metrics had also good performance and they may have better results when adapted to a no-reference model.

1.4 Overview of the thesis

This project is called “Real-Time monitoring and prediction of Internet Video” and may be divided in three sections.
The first section is focused on research in the fields of Internet protocols and video codecs. As the project focuses on video transmission over Internet, we need to acquire a good knowledge in these two fields in order to be able to develop the software this thesis consists on.

In the second section, we have developed software on video quality prediction. In this section, we have to use the knowledge acquired in Internet protocols and video codecs to be able to develop a no-reference method to predict the QoE of the video transmitted over Internet.

In a third section the goal has to be to validate the software by testing it in a real scenario simulation.

This project has been developed at Lund University under the supervision of Maria Kihl. It is part of large European project called IPNQSIS [7] within the Celtic Research and Development program. In this project research is performed towards the development of monitoring systems to study the behavior of Quality of Experience (QoE). These monitoring systems will provide knowledge to assist the design of new and intelligent techniques for distribution of multimedia content offering an acceptable level of QoE.

CHAPTER 2

2 Video Quality Prediction

2.1 QoE: a subjective metric

According the ITU-T Focus Group on IPTV, Quality of Experience refers to *the overall acceptability of an application or service, as perceived subjectively by the end-user*. The opinion of the end-user will lie on the definition of the image viewed. In the figure 1 we can see the same picture with different definition which our eye is capable to perceive. Being defined as a subjective perception, a proper way to measure it should require tests with actual users.

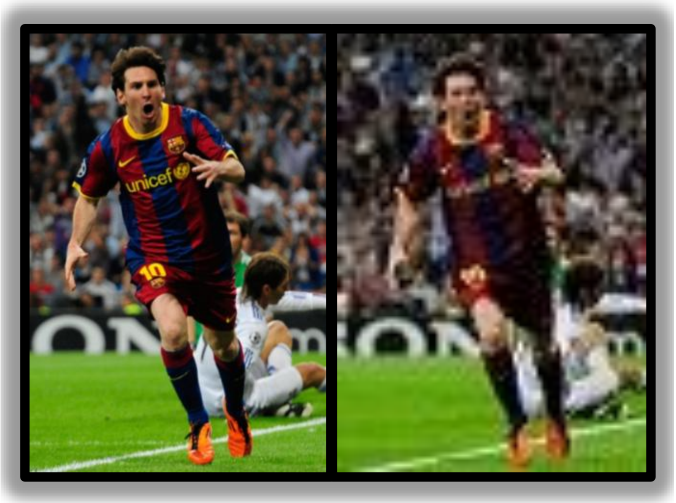

Figure 1: Different perceptive video quality

Since the end-user will qualify the service, all the effects in the end-to-end system are being considered as distortions on the QoE. It includes the quality of the original video (such as bit rate or loss of quality in the compression) and the distortions introduced in the network (such as packet loss, packet reordering or jitter). Also, user perception can be influenced by user expectations ant the context they watch the video.

In order to measure QoE, as said before, tests with actual users are required. The test that is usually used is the Mean Opinion Score (MOS), which consists on a scale from 1(bad) to 5(excellent) which viewers use to rate the video. This is a time-consuming and costly process, so service and network providers work towards finding objective tools [8] that give an accurate estimation of the subjective metrics.

2.2 *Objective metrics*

Objective metrics are tools designed with the aim to estimate the QoE with the best accuracy they can. To do it, they have access to the service information and with it, they obtain objective results.

There are three types of objective metrics, depending on the information they have access to: Full-reference, reduced-reference and no-reference models.

2.2.1 Full-reference models

Full-reference models are models that have access to the original video and can use it as reference. As can be seen in the figure 2, these methods compare the original undistorted video to the degraded video that is received in the customer end and give a value to rate the impact of the degradation. Full-reference metrics compute the quality difference by comparing every pixel in each image of the distorted video to its corresponding pixel in the original video.

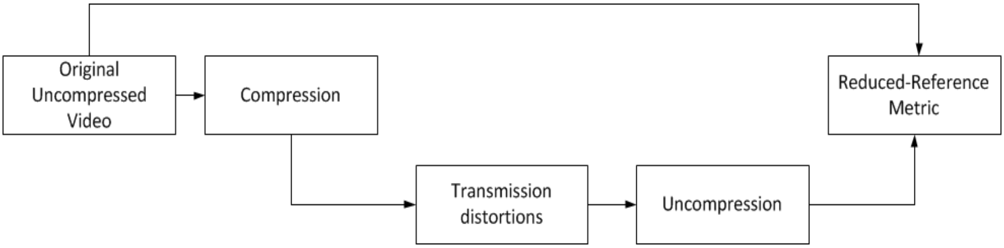

Figure 2: Full-Reference models methodology

Full-reference models provide the most accurate estimation of the QoE, but the techniques used are very complex, heavyweight and have a long computation time. Some of these methods have already been standardized [9].

2.2.2 Reduced-reference models

With the same methodology used for full-reference models, reduced-reference models also base their estimation on a comparison Figure 3 shows us how the methodology is very similar. Even though, the reduced-reference models use a reference video that has already been compressed. Also, the comparison is not done pixel by pixel but in blocks, usually of 8x8 pixels. Working with pixel blocks, these methods are able to estimate many aspects of the video quality, related to the motion of the scenes, delays, power, structure or reconstruction impairments.

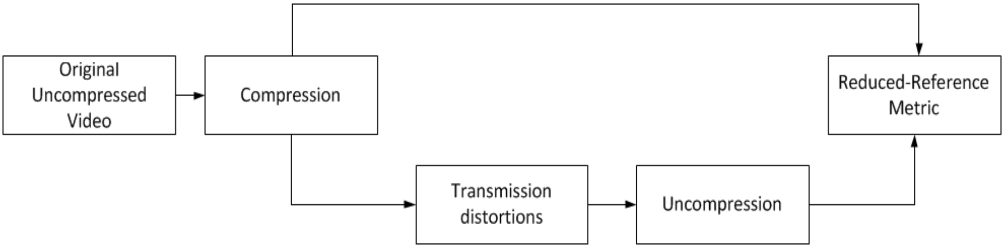

Figure 3: Reduced-Reference models methodology

This fact has two consequences: it makes the QoE estimation less accurate than the full-reference estimation, but on the other hand, reduced-reference methods are faster as the reference is smaller and, in consequence, faster to process.

There are also some reduced-reference models that have been standardized [10]. Within the most used ones we can find PSNR, VQM, SSIM or MOVIE [11] which try to compare both videos in respect of power, impairments or structure.

2.2.3 No-reference models

At last, there are no-reference models. These models do not have access to any reference but they can only use the information they can obtain in the network, generally they use the video stream that goes through the network, as seen in figure 4, for the QoE estimation.

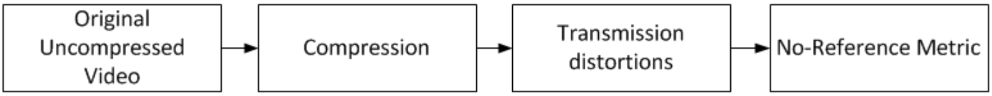

Figure 4: No-Reference models methodology

The methodology used in these methods consists on monitoring QoS parameters in order to get an estimation of the QoE. With that, no-reference models accuracy is lower than full-reference or reduced-reference models, but they are the simplest methods and the only ones that can be run in real-time, so they are the only ones that allow a real-time QoE monitoring. Even if the estimation accuracy is lower than in the other models, these are the only models that we can use in this project. So we need to investigate how a no-reference model is developed.

2.3 Development of a No-Reference model

As we just mentioned, no-reference models accuracy is lower than full-reference or reduced-reference models. Even though, the accuracy has to be maximized. To do that, the method has to be carefully designed to achieve the better estimation of the QoE possible. Starting from subjective methods and through full-reference and reduced-reference methods, the development of a no-reference model requires several tests that lead to the optimum design of the no-reference model. Figure 5 gives us a scheme of the development procedure.

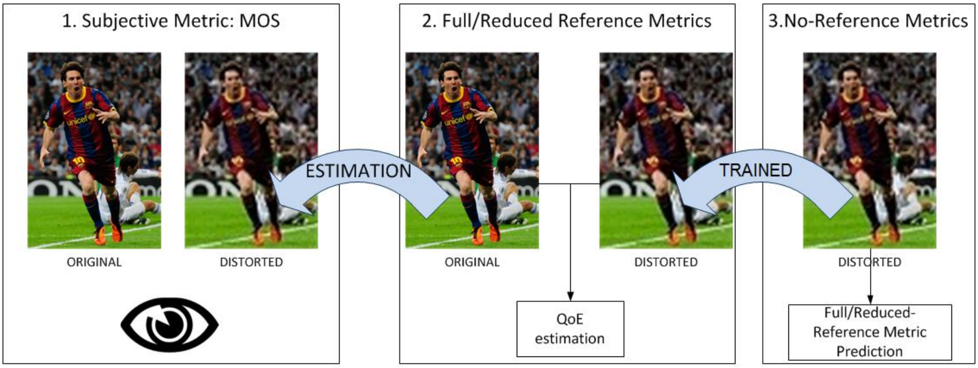
1. Subjective Metric: MOS
ORIGINAL
DISTORTED
ESTIMATION
2. Full/Reduced Reference Metrics
ORIGINAL
DISTORTED
QoE estimation
TRAINED
3.No-Reference Metrics
DISTORTED
Full/Reduced-Reference Metric Prediction

Figure 5: No-Reference method development

The development of a no-reference model can be divided in three parts: QoE and full/reduced-reference tests, assistant full/reduced-reference model choice and no-reference design.

The development of a no-reference model requires a Video Database with several videos with different quality each. First of all, these videos have to be qualified with subjective metrics; this means that the videos have to be displayed to multiple viewers who will rate them using the MOS scale. With that, subjective qualifications are obtained.

After the MOS evaluation, several full and reduced reference methods are applied to the video database. The values obtained are correlated with the MOS values to obtain the method with the highest correlation. This method will be the one that gives the highest accuracy and will be chosen as the model to assist the design of the no-reference model.

As the referenced models estimate the QoE as accurately as possible, the no-reference model has to output values as close to the referenced model as possible. These values need to be obtained as a function of QoS parameters. That is why, in order to find this function, the no-reference model is trained with the referenced model. First, all the values of the referenced model are related to QoS parameters of each video. This relation between the referenced model and the parameters is used to obtain the algorithm that defines the no-reference model, which will have a similar performance with the referenced model due to the relation between their functions.

CHAPTER 3

3 Previous work

This project is the continuation of a long research conducted by Iñigo Sedano under the supervision of Maria Kihl from the LTH and in cooperation with ACREO researchers Kjell Brunnström and Andreas Aurelius. They, together, have been working on different ways to monitor the quality of video over Internet and developed *RQM (Real-time Quality Metric)* algorithm, the no-reference method this project's software this project is based on.

3.1 Full reference video quality metrics

The first step taken to develop *RQM algorithm* was to evaluate some full-reference and reduced-reference objective metrics on three video databases that had been already rated with the MOS scale. The metrics were applied on all of the databases and compared to the subjective ratings using correlation coefficients, and obtaining the results shown in figure 6:

HDTV VIDEO DATABASE UNCOMPRESSED REFERENCE

	Pearson	**Spearman**	**RMSE**	**Outlier Ratio**
PSNR	**0.661**	**0.600**	**0.422**	**0.555**
SSIM	**0.720**	**0.653**	**0.391**	**0.518**
MS-SSIM	**0.727**	**0.664**	**0.386**	**0.518**
VSNR	**0.629**	**0.511**	**0.438**	**0.592**
VQM	**0.840**	**0.782**	**0.305**	**0.370**

Figure 6: Correlation coefficients

With that, VQM was chosen as the appropriated reduced-reference metrics to assist the development of *RQM algorithm* because of being the method with the highest correlation coefficients, as we can see in figure 6.

Video Quality Metric (VQM) software provides methods to estimate how people perceive video quality using algorithms of comparison between the original and the received videos [12]. VQM operates on the original and received video files and reports video calibration and quality metric results such as: temporal and spatial registration, spatial scaling, gain/level offset and objective video quality estimation. The latter is the interesting result for this research. The quality is estimated on a scale from zero to one where zero means that no impairment is visible between original and received videos, and one means that the video has reached the maximum impairment level. This software also provide tools for detecting dropped video frames [13] and estimating variable video delays [14].

3.2 No reference video quality metrics

As we said before, the last step to obtain a no-reference model was to obtain an estimation of the VQM values as function of parameters of the network calculable using the data stream and use this relation to obtain the RQM algorithm. Each value of VQM obtained was related to the packet loss rate, denoted p, and group of picture, denoted I as variables. Using the MATLAB function *nlinfit* [15] the coefficients of the relation were obtained:

$$\boxed{VQM = b_0 + b_1 \cdot I^3 + b_2 \cdot I^2 + b_3 \cdot I + b_4 \cdot p^3 + b_5 \cdot p^2 + b_6 \cdot p}$$

This relation between VQM and packet loss and group of picture was used to obtain the algorithm of the no-reference method, which had to obtain values close to the VQM values. The algorithm obtained is shown above, and figure 7 shows the closeness between VQM values and RQM surface.

$$\boxed{RQM = -0.16 - 0.0001 \cdot I^2 + 0.0064 \cdot I + 0.0003 \cdot p^3 - 0.0092 \cdot p^2 + 0.1106 \cdot p}$$

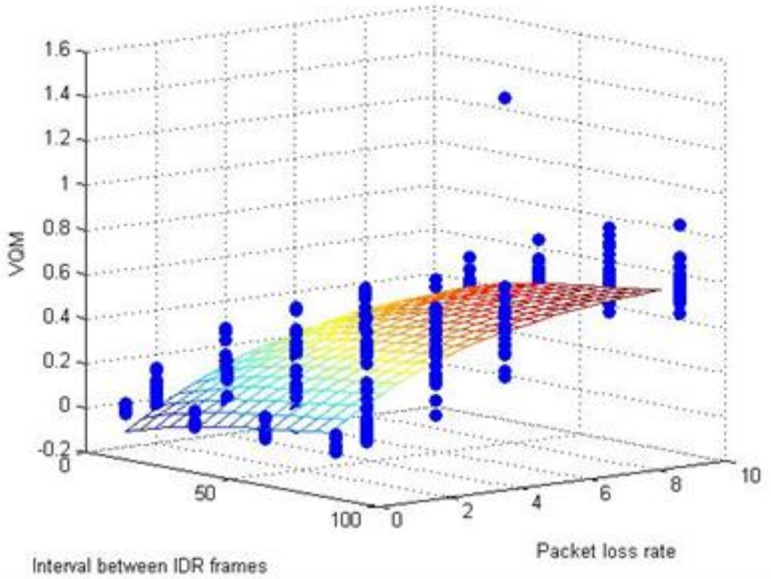

Figure 7: VQM values and RQM surface

Once *RQM algorithm* was developed, it was also applied on the video databases and validated being compared with the subjective ratings by the same correlation coefficients that the full-reference models.

CHAPTER 4

4 Theory on real time video streaming

In order to monitor the QoE of the Internet video services, we are going to use *RQM algorithm*. As mentioned in the previous chapter, this no-reference method obtains RQM as a function of the packet loss and the group of picture of the video:

$$\boxed{RQM = -0.16 - 0.0001\cdot I^2 + 0.0064\cdot I + 0.0003\cdot p^3 - 0.0092\cdot p^2 + 0.1106\cdot p}$$

Now, the two parameters we need to calculate are packet loss and group of picture. The first one can be calculated through the information given by the protocols headers and the group of picture through the video codec information.

So in order to be prepared to find a way to estimate these two parameters and obtain the RQM value, we have to acquire knowledge about RTP protocol and H.264 video codec. With that, we will be able to develop the estimation algorithms of the software.

4.1 Real-Time Transport Protocol

This protocol is used in the application layer and usually works above UDP. It provides end-to-end network transport mechanisms suitable for real-time multimedia transmissions. Such as payload type identification, sequence numbering, timestamping and deliver monitoring. It requires the use of a control protocol (RTCP) to monitor the data delivery.

All the mechanisms are provided in a lightweight header shown in the figure 8.

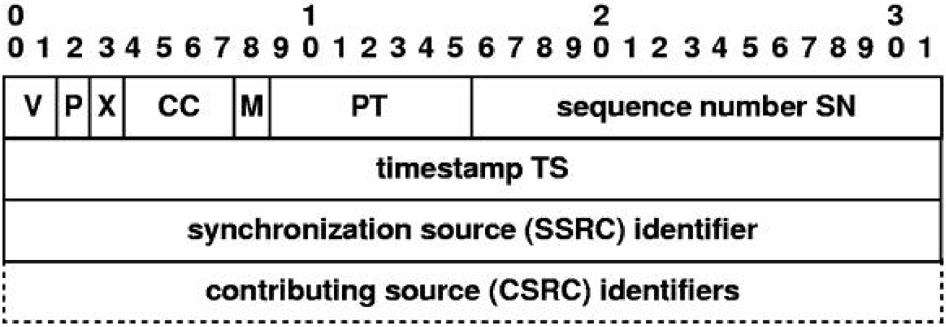

Figure 8: RTP header

It is a 12 bytes long header with possible extension fields (CSRCs) if the stream goes through a mixer.

The fields in the header provide information, not functions [16]. This means that is not a task for RTP but for the application in the receiver to use this information to improve the quality of the transmission.

4.2 H.264 video codec

H.264 can encode videos with High-Definition with lower bitrates than the rest of codecs. Actually, it has been proved to save up to 40-50% bit-rate providing equivalent quality that MPEG-2[17]. H.264 technology is currently used in Blu-ray Discs, HDTV, HD recording, mobile devices (including iphone, Samsung, PSP…) and online high-quality content.

The main techniques that provide this improvement are related to the motion compensation, DCT algorithm, frame reconstruction or frame partition [18].

H.264 encodes the video with three different types of frames: IDR, P and B frames. IDR frames are used as reference for the reconstruction of other frames and also as a hook for random access to the video. The compression is very low, so the frame can be auto-reconstructed.
P frames (P for predicted) need previous IDR frames to be reconstructed. The coded frame does not have information enough to be auto-reconstructed. It contains image and motion vector displacements information.

At last B frames (B for Bidirectional predicted) require previous and future frames to be decoded, the reference frames can be both IDR and P. The compression is very high in this type of frames and they require more references to be reconstructed.

P and B frames are high-compressed because they contain motion vector displacements information. This information relates to the difference between two adjacent slices and is much less amount of information than picture information. Figures 9 to 12 show the mentioned information: two adjacent frames, the difference between them and the displacements created.

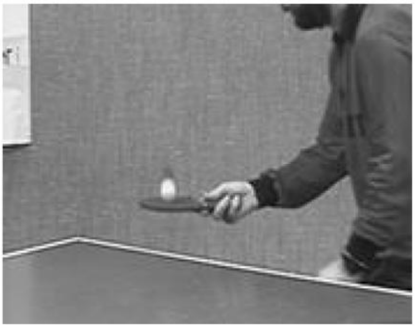

Figure 9: Frame #1

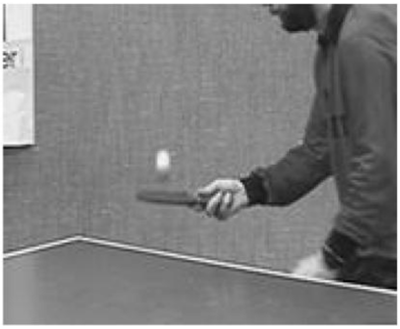

Figure 10: Frame #2

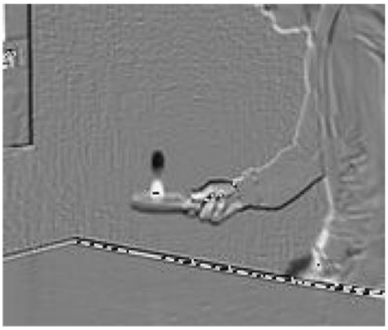

Figure 11: Frames difference

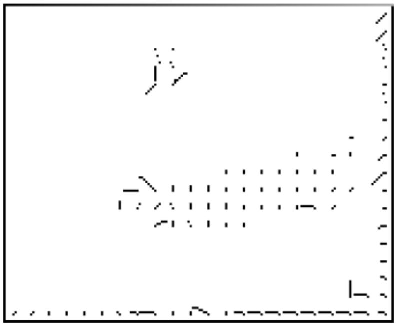

Figure 12: Motion vector displacements

The interval between IDR frames is also known as group of picture (GoP); one of the two parameters used in *RQM algorithm.*

Figure 13 illustrates the reconstruction process used by the H.264 codec with the different frames encoded. In the figure, the arrows indicate the direction of the prediction and shows us how P frames are reconstructed using the reference of the previous IDR frame and B frames use previous and future IDR and P frames to be reconstructed.

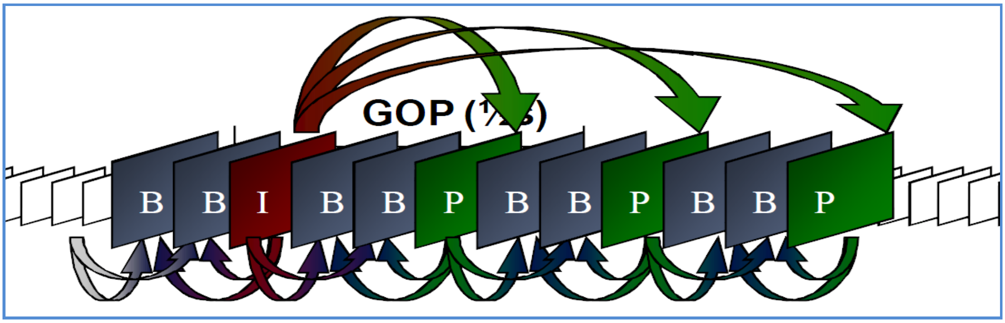

Figure 13: Video coding system

H.264 generates IDR frames to be a reference for frames reconstruction when P and B frames are not capable to do it. Also, IDR frames are used to create points of access to a random access to the video. Therefore, they are not going to appear in a regular interval, but this interval will be variable.

One last feature of the H.264 is the encoding process division in functional layers: Video Coding Layer (VCL) and Network Abstraction Layer (NAL). The VCL includes the signal processing functionalities of the codec (transform, quantization, motion compensated prediction, etc.). The VCL encoder outputs slices: a bit string that contains the macroblock data of a number of macroblocks and the information of the slice header. The NAL encapsulates the slice output of the VCL into the NAL unit payload. The NAL unit is formed with a 1 byte header and the payload byte stream. NAL main goal is to provide “network friendliness” to enable simple customization of the use of the video data and to facilitate the mapping of the video data to the transport, including RTP, dividing the data in NAL units. In the RTP case, there are different payload formats depending on the type of NAL unit transmitted.

4.3 RTP payload format for H.264 Video

As we have just seen, the RTP payload format can change depending on the payload type and the RTP profile. For H.264 video the payload type is dynamic. That is because RTP payload format allows for packetization of one or more NAL units in each RTP payload, as there are three different payload structures: single NAL unit packet, NAL unit aggregation packet and fragmented NAL unit packet. As their name indicates it, the payload of these packets will be a single unit, several units or a partition of a NAL unit respectively, depending on the information being transmitted and the size of the slice that the NAL unit contains. The NAL header provides information about the type of NAL unit, in the case of an aggregation NAL unit the different units are separated by a prevention byte, but aggregation NAL units do not concern us in this project as they cannot carry frames information. And at last, in the case of fragmented units there is one extra 1 byte header generated to allow the reconstruction of the slice. Figure 14 shows the structure of each type of NAL unit and the headers.

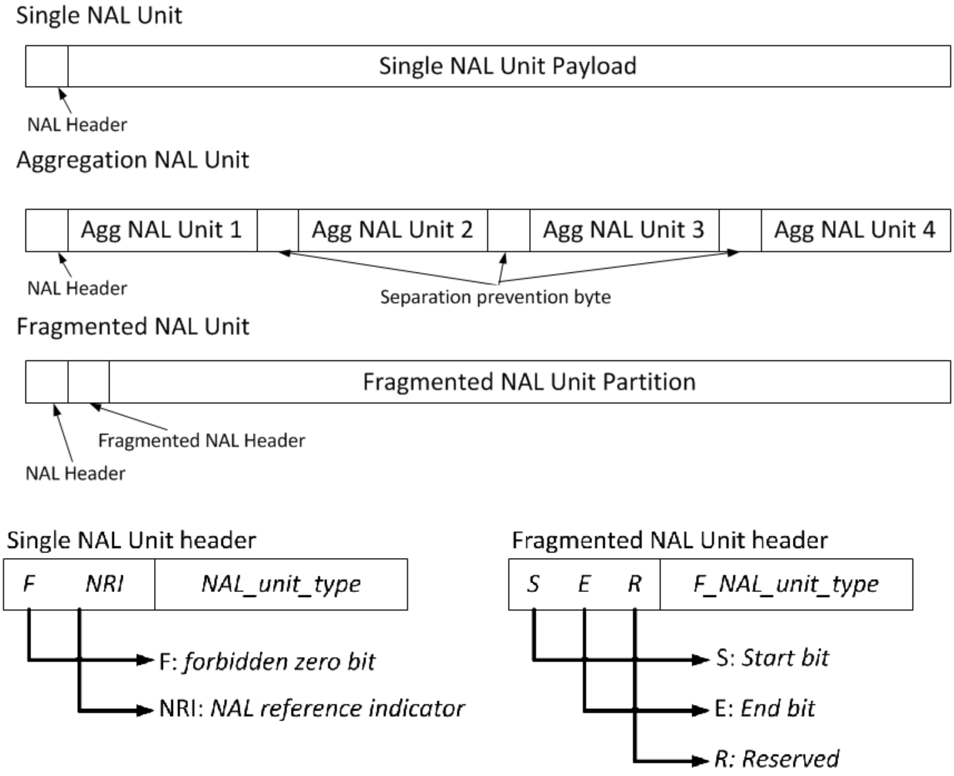

Figure 14: NAL Units structure

CHAPTER 5

5 Video Quality Prediction Software

The goal of the developed software is to receive a degraded video coded with H.264 and transmitted using RTP protocol and calculate the RQM according to the no-reference model, which basically implies to calculate packet loss rate and GoP. With that, the software can be divided in three parts: UDP connection, packet loss estimation and group of picture estimation, for at last use the RQM algorithm, as seen in figure 15.

Then, in order to validate the software, a net demo and video players have to be studied and used.

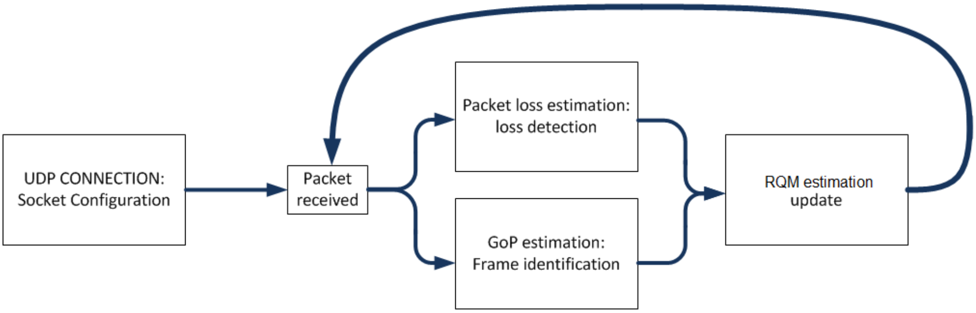

Figure 15: Software operation

5.1 UDP Connection

The first step that has to be taken is to receive the video. As it is received through a UDP connection, we need to establish a socket connection with the video sender machine.

UDP transmission uses datagrams. This means that a pair of sockets doesn’t need to connect and accept each other before beginning the transmission. All we need to do is to open a socket, configure it as a datagram socket and bind it to an address. The address consists on an IP address and a port number that the socket will use to communicate remotely to other machines. The IP address will be set to 10.0.0.2 and the UDP port for RTP uses to be the number 5004, but we will keep it indeterminate and set it at the moment of the simulation. We just need the sender to transmit the video stream to the same port we configure in the software.

Once the socket is bind to an IP address and a port number the socket is set, open and ready to send and receive. In this situation, we'll use it to receive the video packets and store them in a buffer.

5.2 Packet loss estimation

Once the video is being received, we need to calculate the packet loss and the group of picture with every packet received.

To start with the packet loss rate, we need to use the RTP header. From there we can get information about the packets that have been lost and we'll be able to calculate the rate. As seen in the last chapter, there's a two bytes long field in the header called Sequence Number that includes a number that increments by one for each RTP data packet sent by a single source. We can use this field to realize when a packet has not been received. The procedure followed is: to keep the sequence number of the last packet that has been received and to compare it to the sequence number of the packet following. We can keep the track of how many packets are being lost and how many are being received with two counter variables. Using this variables to know the percentage of packets lost over the overall of packets (received + lost) we'll get the packet loss rate. Figure 16 illustrates the base of the algorithm: sequence number counting.

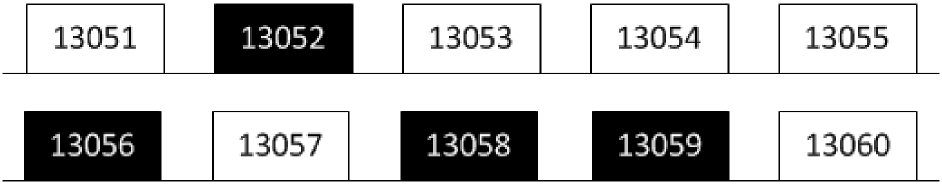

Figure 16: Packet loss estimation base

This estimation will give us the packet loss rate in real-time, as it will change every time a packet is received. That's exactly what we want, because in this way, the software is sensitive to bursts of packets lost and other spontaneous critical situations. If the packet loss would be estimated as an average rate, it would be less sensitive to these situations.

Here, we show a simplification of the algorithm used:

```
packets_received = packets_received + 1
if SN > (last_SN+1)
        packets_lost = packets_lost + SN – (last_SN + 1)
packet_loss = packets_lost / (packets_received + packets_lost)
```

Although, this method to estimate the packet loss rate are correct if used in a relatively short RTP session. If the session lasts long the estimation may take an average value and lose sensitivity to critical situation. Unfortunately, this is the only method we have to estimate the packet loss and actually it is good enough for an average video streaming session. For longer sessions mechanisms can be developed to keep the estimation sensitive to critical situations.

Also, we have to make sure the packets we’re receiving are from the video source we want to receive it from. This cannot happen in the simulation, but in a real scenario we can receive more than one video stream in the same port and we have to be able to separate them for the QoE estimation. That can be done checking that the value of the SSRC field of the RTP header the same that the last packet of the correct video received. The SSRC is a 4 bytes long field that distinguishes uniquely the video sources.

5.3 *Group of Picture estimation*

For the estimation of the group of picture we have to know the interval between IDR frames. The easiest way to do that is to recognize the IDR frames and to count how many non-IDR frames are between two adjacent IDR. To do that, we can use the information given to us by the H.264 header.

In a similar way as we have done with the packet loss rate, we will have to use some fields in the header to conclude that we have received either a packet containing part of an IDR frame or a non-IDR one. The method created to detect these types of frames is explained below and illustrated in figures 17 and 18.

As seen in the last chapter, when an H.264 video is transmitted with RTP protocol, the payload has a dynamic format depending on the NAL unit type, and that single and fragmented units can carry an IDR frame, but not aggregation units.

As said before, we need to identify the IDR frames. These will be in a particular type of single NAL unit or fragmented unit payload. The NAL unit payload type is specified in the NAL unit header, in a 5 bits field named *nal_unit_type*. With this field we can identify the payload of the NAL unit and detect: single NAL units carrying an IDR frame, single NAL units carrying a non-IDR frame and fragmented NAL unit.

Identifying the packets with value 5 in this field we obtain the single NAL unit packets that carry a slice of an IDR picture. As the whole frame will be in the payload, no more work needs to be done to detect the IDR frame.

On the other hand, a value 28 in this field implies the packet contains a fragmented unit. Once we know the packet being analyzed contains a fragmented NAL unit, the next step is to know whether the NAL unit has an IDR frame or not. To do it, fragmented units have an extra header to provide information to merge the fragments but it also provides the *nal_unit_type* field from the H.264 header that in this case also describes the NAL unit payload type. In the same way that in the single NAL unit payload, a value equal to 5 means an IDR frame is transmitted in that packet. Once we know we have a fragmented NAL unit carrying an IDR frame we can use the *start_bit* and *end_bit* fields in the fragmented unit header to identify the start and end of the frame in packets.

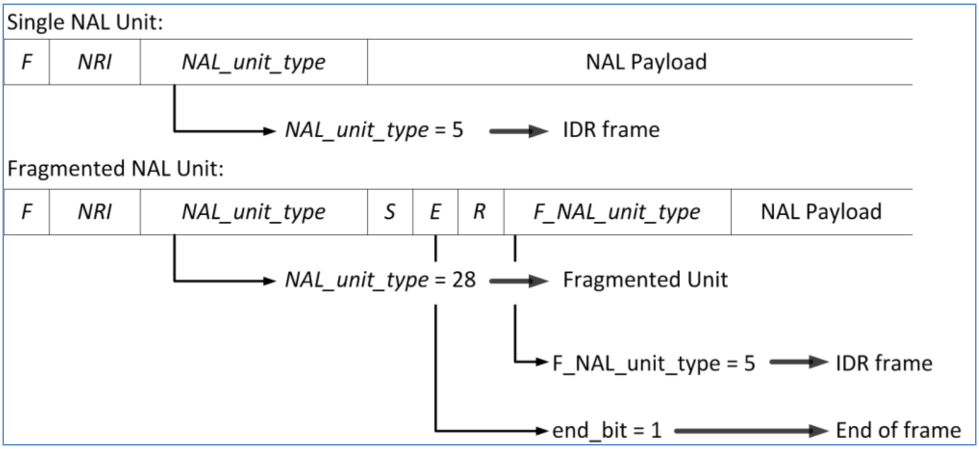

Figure 17: IDR frame detection

Once we know how to identify IDR frame carrier packets, we just have to count the non-IDR frames received between two adjacent IDR frames. It can be done in a similar way it has been done to detect IDR pictures, depending on the type of NAL unit format we have: single or fragmented unit.

Single NAL unit payload containing a non-IDR frame can be distinguished with the *nal_unit_type* field of the header; its value is 1 in this case. Just like the IDR frames case, the whole frame is in the payload.
When frames are in a fragmented unit, every packet transmits a part of it. Once again, what we need to do is to find out what is the size of the picture in packets. Using the same strategy we used before, using the *start_bit* and *end_bit* fields of the fragmented unit header, we can find it out.

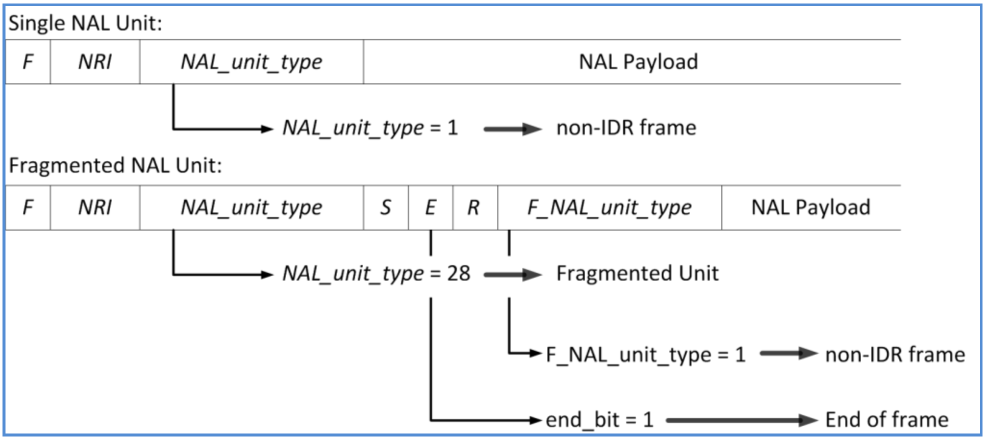

Figure 18: non-IDR frame detection

Once we are able to detect both IDR and non-IDR pictures, the algorithm has to count the non-IDR frames being transmitted. When we receive an IDR frame take the counter value as the last group of picture and reset the counter.

```
if(new_frame = IDR):
            gop = counter
            counter = 0
else:       counter = counter + 1
```

5.4 Other software

To run the simulation there is more software that has been already implemented, but we need to use or modify. This software is a net demo and two video players.

5.4.1 DemoGUI

DemoGUI is software developed by a former student of the LTH University and it is designed to emulate a network and its effect on a video transmission. The tool functions are very complete and it is able to stream video with many protocols and distorting it in different ways.

We use the tool because it provides RTP transmission and it is able to add packet loss to the transmission, but we need to modify an important detail of the transmission: the video needs to be coded in H.264. The program is written in Java, and all we need to do is to change the line of the code where it runs the VLC video player and include the codec modification.

5.4.2 VLC

VLC has two functions that we can use in this project: video streaming and video codification. It allows the streaming of video over several protocols, so we will send a video to our software over an UDP/RTP connection.
Also, it can convert any video to different codecs than the original ones. In the case of the H.264 codec, it can convert videos to H.264 and modify its parameters, as bit rate or group of picture. That can be used to do some simulations to validate the group of picture estimation part of the software.

5.4.3 ffplay

VLC has one problem that makes it useless to receive the video we stream and forces us to find another player to reproduce the video sent and distorted by the net demo. VLC is not programmed yet to receive a H.264 video over RTP. This is due to the fact that, as we have seen before, the RTP payload is special for the H.264 video.

Instead, we have found a very complete, cross-platform solution to record, convert and stream audio and video. It is named ffmpeg [19] and it’s the most complete software for video coding, editing and streaming. We can use it to display the distorted video.

CHAPTER 6

6 Results

Once the software is developed, we need to run different simulations in order to validate the performance of the software. As the software estimates separately the packet loss and the group of picture, these two parts can be tested separately as well and then test the complete software.

6.1 Simulation System

At this point the software is developed and all what needs to be done is to validate it. To achieve this, we have to run it in the demo system we have available.

The system is formed by the computers connected by an Ethernet cable and configured, as seen in the figure 19, to be able to transmit data through this connection. In the first computer we have the DemoGUI platform, already modified to stream the H.264 video and in the second computer we have our prediction software and the ffplay to reproduce the received video.

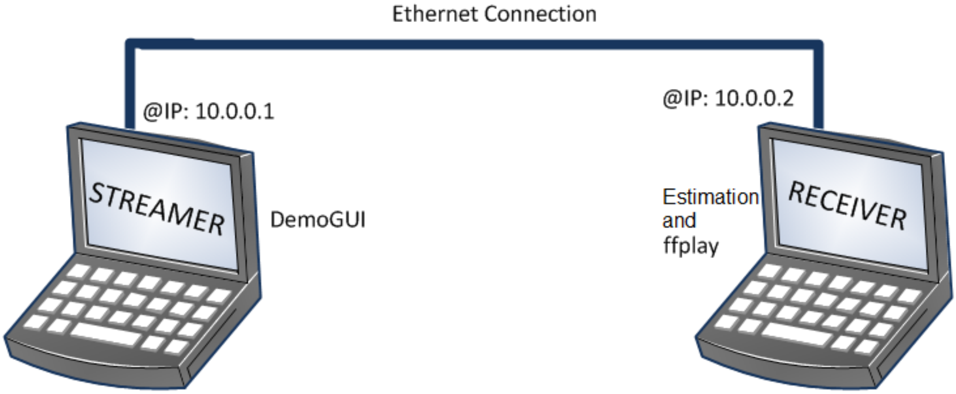

Figure 19: Simulation System

With this system, we can test the predictor software and analyze its performance. This analysis may start with a packet loss estimation test and a group of picture estimation test without packet loss to test the precision of the estimations. Then, we could study the affection of packet loss on the

GoP estimation. At last, we can compare the model developed with the model found in the previous work.

6.2 Packet loss estimation test

6.2.1 Preparation

To test the performance of the packet loss estimation part of the software we can use the DemoGUI platform to stream the video to the receiver computer. We have to configure it to send it to the @10.0.0.2 port 5004 (default RTP port).
In order to test just the packet loss part, we need to modify the code so it just outputs the packet loss rate instead of the RQM value so we can take it and use this output to validate the performance of the software.

The packet loss simulations will be done fixing the rate to 5%, 10%, 15%, 20% and 50%. Even if they are not realistic values, it will be useful to prove the software works in any rate.

6.2.2 Results

By capturing the packet loss estimation in a H.264 video of 3:57 minutes in the mentioned situation, we obtain the graphic shown in the figure 20.

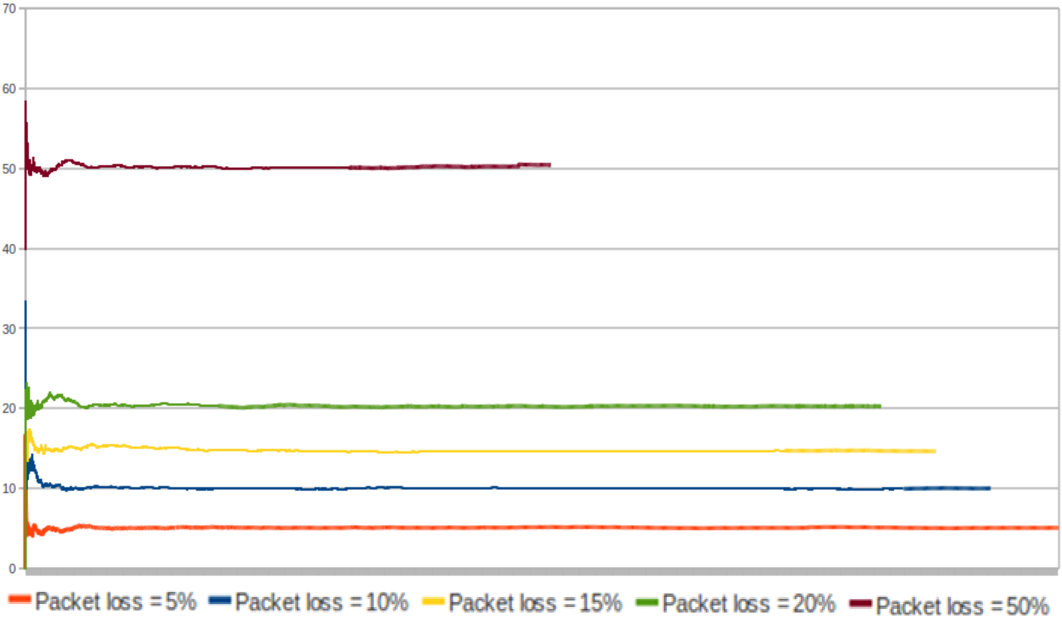

Figure 20: Packet loss simulations

Observing the figure 20 we can conclude that the packet loss estimation outputs a correct value of the rate, as the average value is clearly around the rate fixed with the DemoGUI platform. Even so, the first seconds of the video the prediction goes through a transitory zone. That is due to the fact that lost packets have more influence in our algorithm when few packets have been received. We can also take a look at the statistics of the estimation in the table 1:

	Packet Loss = 5%	Packet Loss = 10%	Packet Loss = 15%	Packet Loss = 20%	Packet Loss = 50%
Mean	5,1176	10,0906	14,7777	20,3406	50,1754
Variance	0,0318	0,2069	0,0888	0,0958	0,1216
StDeviation	0,1784	0,4549	0,2979	0,3095	0,3487
Max value	16,6666	33,3333	20	23,2558	58,3333
Min value	4,0964	9,764	9,0909	10	40
Max error	11,6666	13,3333	5,9091	10	10
Rel. Error	2,35%	0,91%	-1,48%	1,70%	0,35%

Table 1: Packet loss statistics

We can see how all average values of the estimations are near the real rate. Although, we see how the variance is quite large. This is most likely because of the transitory zone. If we see how the statistics are in the stable zone, we can see the performance is much better (consider the stable zone after 1000 packets):

	Packet Loss = 5%	Packet Loss = 10%	Packet Loss = 15%	Packet Loss = 20%	Packetloss = 50%
Mean	5,1131	10,0476	14,7669	20,3439	50,1747
Variance	0,0043	0,0059	0,03613	0,0461	0,0381
StDeviation	0,0659	0,0767	0,19	0,2146	0,1951
Max value	5,4081	10,6645	15,5273	21,9774	51,1022
Min value	4,589	9,764	14,3712	19,9549	49,0457
Max error	0,411	0,6645	0,6287	1,9774	1,1022
Rel. Error	2,26%	0,48%	-1,55%	1,72%	0,35%

Table 2: Packet loss statistics (stable zone)

We can see how the performance is improved in the stable zone. Even if it is not absolutely precise, we can see how variance, standard deviation and maxim error are considerably decreased. So it proves that the stable zone gives a much better estimation than the transitory. We can see how the mean is almost the same, so we can say that even if the transitory zone is more random, its mean is still good enough.

The deviation we still perceive in the stable zone could be related to the random packet loss generation by the DemoGUI platform. To investigate this behavior we can run the software twice with the same fixed packet loss rate and observe whether the performances differ or are equal.

We test it with the packet loss rate fixed to 50% and obtain the figure 21.

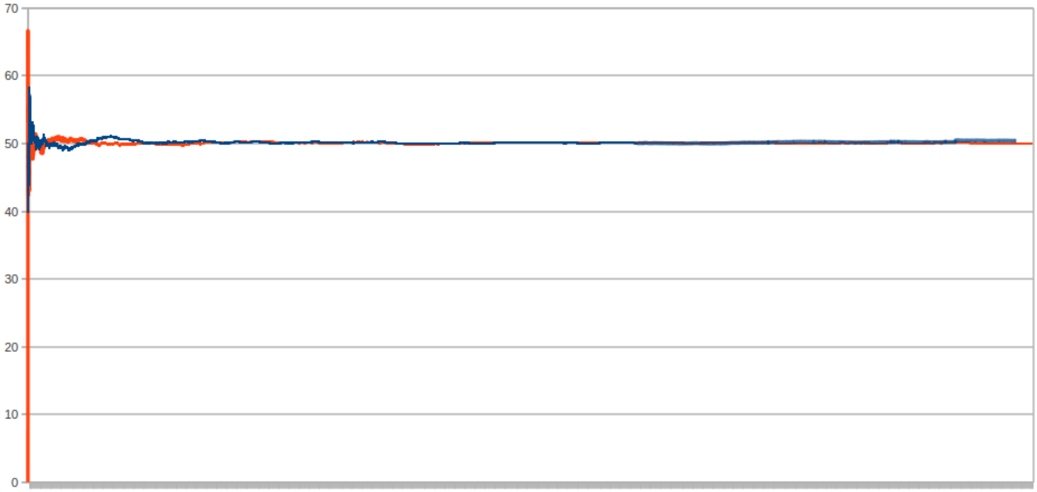

Figure 21: Packet loss = 50% two different simulations

As suspected, DemoGUI random packet loss generation can introduce deviations in the rate estimation. We can see it in the transitory zone; clearly different in both cases, and in the rest of the estimation, where we see, not that clearly, different behavior with thinner or thicker lines in different areas. To make it more clearly we can see the statistics in the transitory (first 1000 packets) and the stable zone in the table 3:

	Simulation 1 (Transitory)	Simulation 2 (Transitory)	Simulation 1 (Stable)	Simulation 2 (Stable)
Mean	50,2085	50,0591	50,1747	50,09158
Variance	2,334	6,5437	0,0381	0,0129
StDeviation	1,5277	2,5581	0,1951	0,1135
Max value	58,3333	66,6666	51,1022	50,766
Min value	40	42,4242	49,0457	49,7136
Max error	10	16,6666	1,1022	0,766
Rel. Error	0,42%	0,12%	0,35%	0,18%

Table 3: Different simulations, same packet loss rate

The statistics information matches what we see in the graphic; both simulations have different performance; all the statistics differ both in the transitory and the stable zone.

6.3 Group of Picture estimation test

6.3.1 Preparation

As we saw above, group of picture (GoP) is a parameter that may be variable during a video encoding. Even if we would try to fix it using the VLC, the codec won't make it fixed as IDR frames are needed in determinate moments, such as high motion scenes or scene changes.

So in order to test the group of picture estimation, the first thing we can do is to use the VLC to stream a video without packet loss. Using VLC we can fix an interval the parameter will have to be within. The VLC will stream the video after coding it with H.264 codec and will generate IDR frames within the specified interval. We have to modify the code of the software so it outputs the group of picture estimation instead of the RQM value.

For the simulation, we use the intervals: 1-50 frames, 50-100 frames, 100-250 frames and 25-250 frames.

6.3.2 Results

We use the same video used in the packet loss test, but we stop it after 2 minutes. The results obtained are shown in the figures 22 to 25.

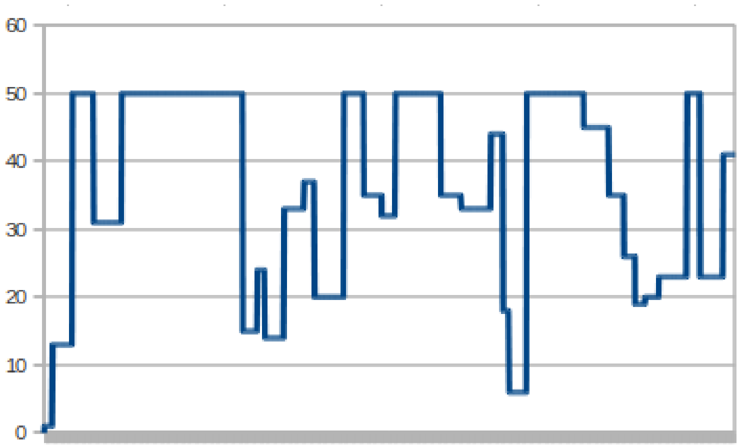

Figure 22: GoP between 1 and 50 frames

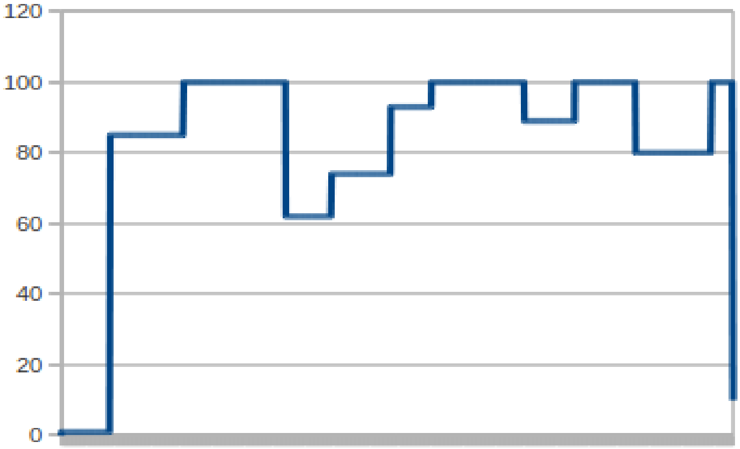

Figure 23: GoP between 50 and 100 frames

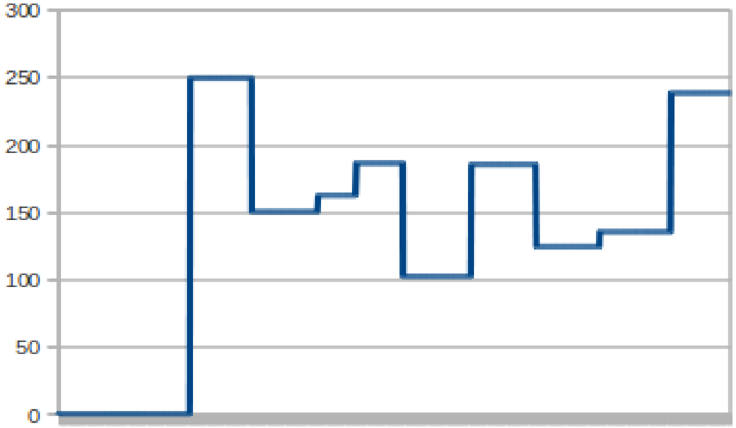

Figure 24: GoP between 100 and 250 frames

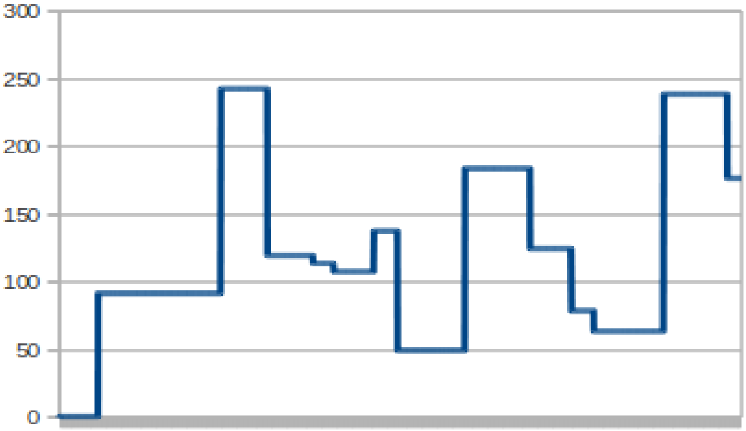

Figure 25: GoP between 25 and 250

In figures 22 to 25 we can see:

- The margins of the interval are respected in all of the simulations.
- The codec tends to take the largest group of pictures possible. That is because a large interval between IDR frames needs a lower bit rate.
- Although, we can see in figures 24 and 25 how sometimes the codec takes short group of pictures. This could be due to a scene with high motion that may require a larger number of references to be reconstructed.
- We can also see the random performance of the codec when it comes to generate IDR frames, which are different in every simulation.

One more experiment we can do is to see how does packet loss affect the GoP estimation. As the estimation algorithm uses data from the packets, the loss of packets could make it miscalculate the GoP. To test it, we have to use the DemoGUI platform again. As it uses the VLC to send the video, the video codec configuration applies here, too, including the GoP range. So if we send the video with a 10% packet loss and a GoP range of between 25 and 250 frames, our software outputs the GoP estimation shown in the figure 26.

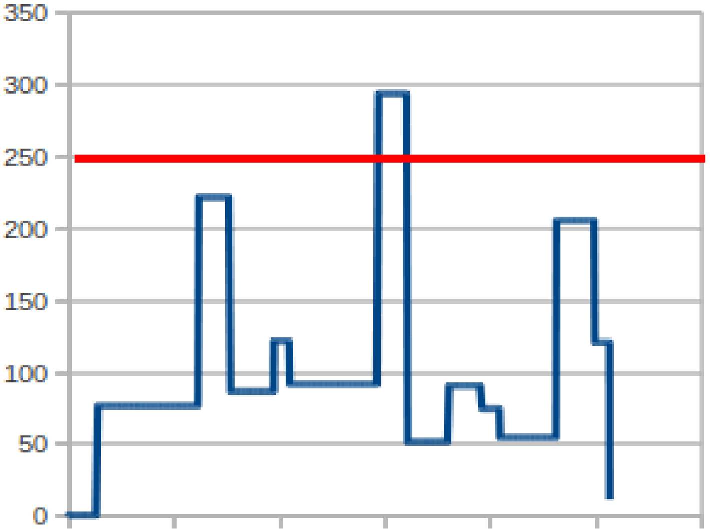

Figure 26: GoP estimation error

As we suspected, packet loss affects the estimation. This can be seen in the figure 26 as part of the estimation is out of the range.

6.4 RQM values comparison with algorithm

6.4.1 Preparation

As we have seen in the packet loss and group of picture estimation tests, the algorithms work fairly well but it is not absolutely precise. Now we can see how this error affects the RQM estimation, and compare it to the one should be the correct value. In this case, the software output will have to be the RQM value, but also its corresponding GoP and packet loss estimation.

To do it, we will run multiple simulations using the DemoGUI to add different packet loss rates in each simulation and we will not fix a range for the group of picture, as we have already seen the performance of the GoP estimation. Instead, we will stream a video that is already coded with H.264. Packet loss rates are fixed to 0.1%, 1%, 3%, 5% and 10%.

6.4.2 Results

To show the results of this test, we will use MATLAB to show a 3D graph with the values of RQM as function of the packet loss and group of picture values obtained in the simulations. In the same graph, we will show the surface defined by the RQM algorithm.

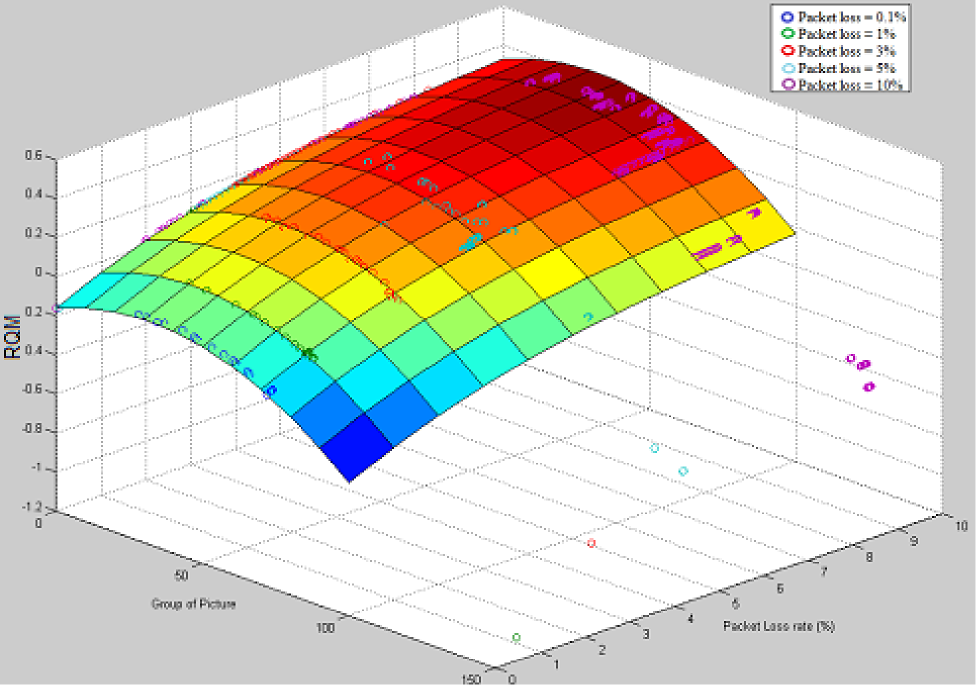

Figure 27: RQM(packet loss, GoP)

The figure 27 shows how almost all of the values obtained in the software are contained in the surface of the model. But we see scattering in both axes, this scattering can be seen more clearly with the zoom of figure 28.

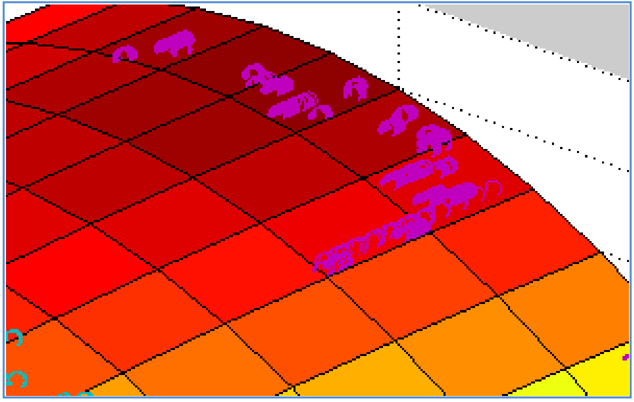

Figure 28: Scattering in the RQM function, both axes

To take a better look at it, we can see the RQM as function of GoP and as function of packet loss separately to see the impact of the errors in the parameters estimation on the RQM estimation.

We can first take a look at the axis of the GoP:

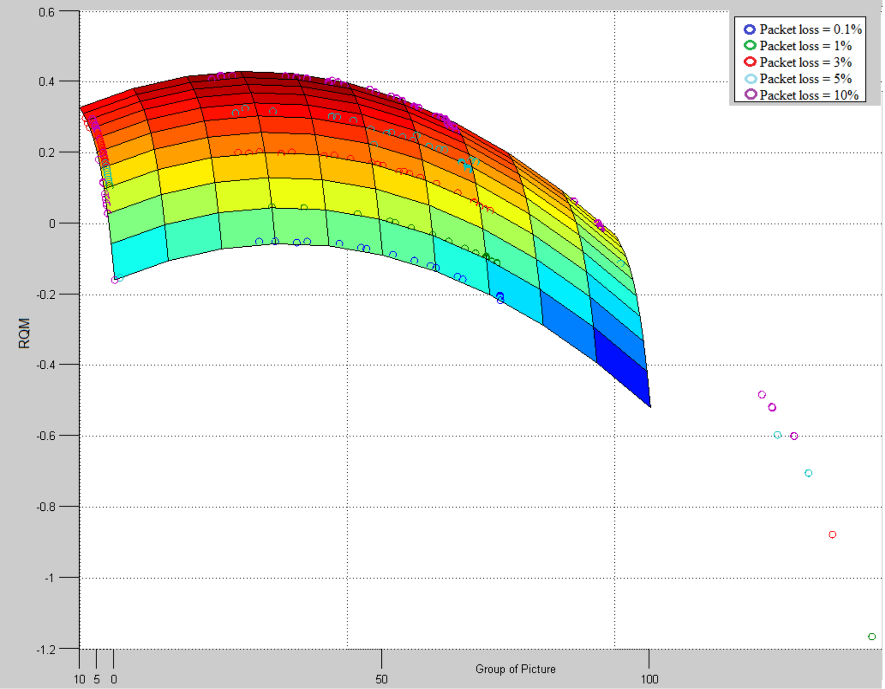

Figure 29: RQM(group of picture)

With the vision of figure 29 we can see how the simulations with high packet loss rates (purple and clear blue) output wrong values for the group of pictures that cause values for RQM out of the functions area.

With the zoom of the figure 30, we can see the scattering in the GoP axis:

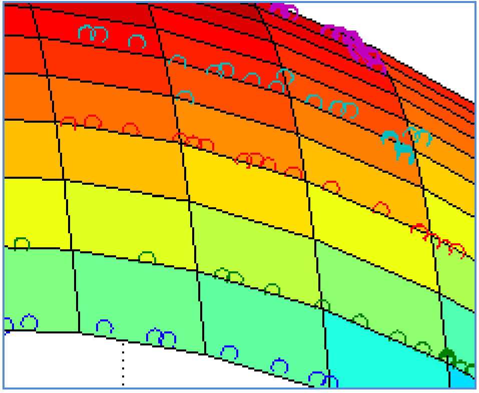

Figure 30: Scattering in group of picture axis

With the zoom of the figure 30 we can see how the values in this axis are more scattered in the simulations with higher packet loss. That is because, as we saw previously, packet loss has an impact on the group of picture.

We can also take a better look at the influence of the packet loss estimation deviation by turning the graphic to see the RQM(packet loss) function.

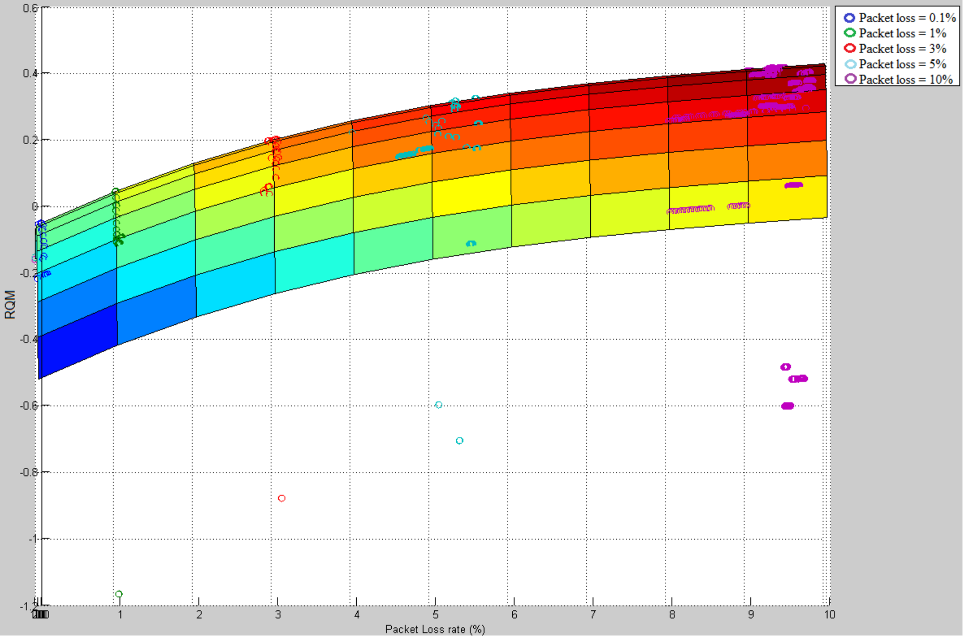

Figure 31: RQM(packet loss)

In the figure 31 we see clearly how the higher rates cause larger deviations in the RQM values; the points in the first simulations (left side of the figure 31) are much closer to the correct rate, but the last simulations, with higher rates, have their points much more scattered. Also, we can see points out of the model surface due to group of picture estimation errors. But we have to remember these errors are caused by packet loss as well, as we can see many more points out of the surface with the higher rates simulations.

We can zoom again to see how scattering is much lower for the low packet loss rates simulations (figure 32) than for high rates ones (figure 33).

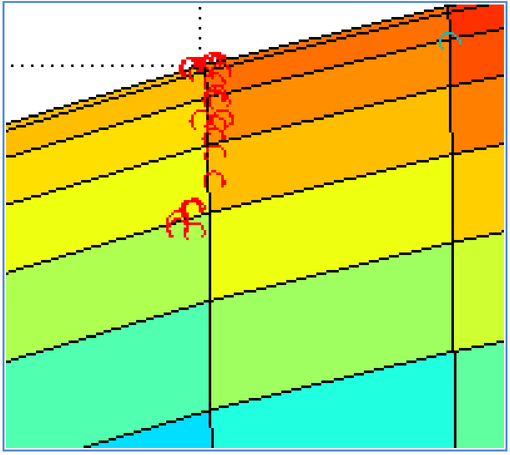

Figure 32: Scattering in low packet loss rates

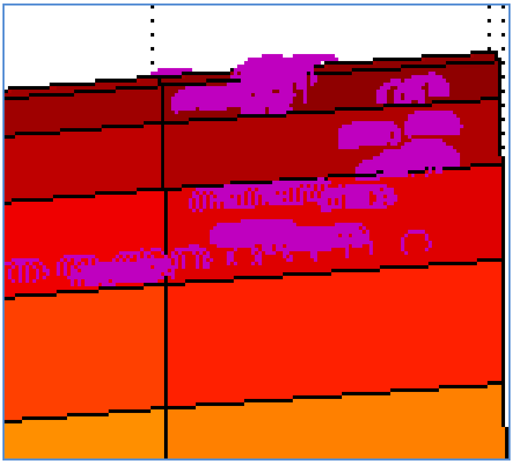

Figure 33: Scattering in high packet loss rates

Finally we can show the results of the simulations from a new angle and see the spread of the values obtained on the surface of the model.

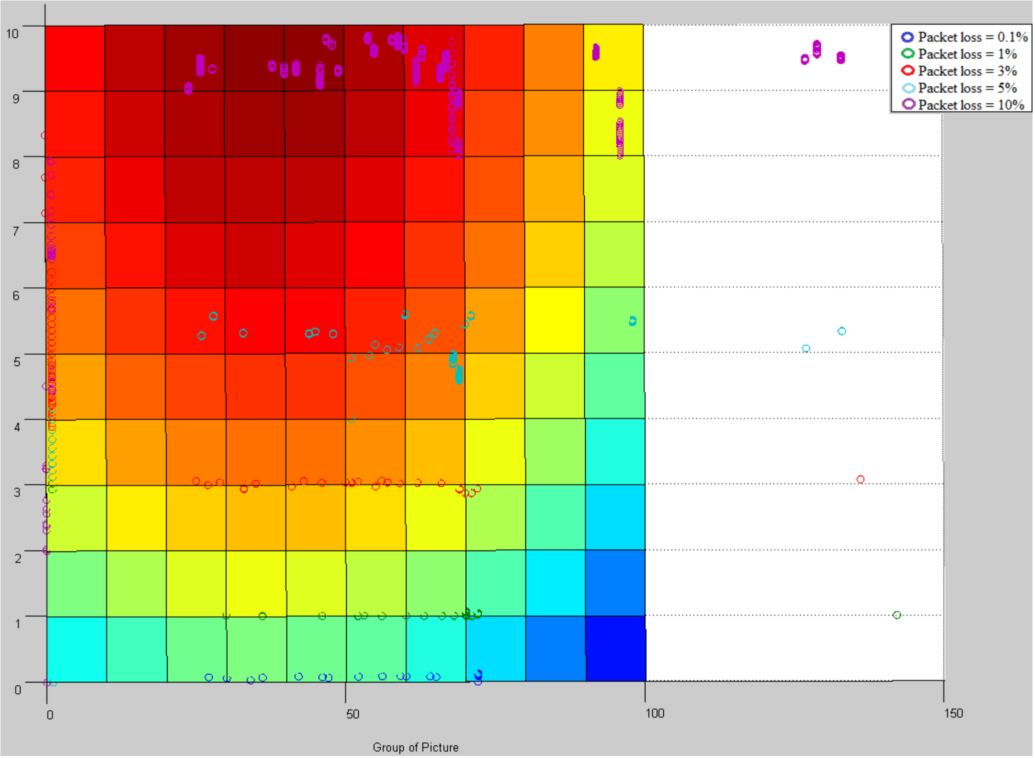

Figure 34: RQM(packet loss, group of picture) on the model surface

In the figure 34 we can see that group of picture errors cause wrong RQM values out of the surface. Even though, group of pictures errors are also generated by high packet loss rates. It is proved by the fact that dispersion is larger in both dimensions as higher is the packet loss rate.

We can also see how packet loss values are less scattered than group of picture and are always around the fixed value. This is because group of picture is not fixed, and can take any value in the interval. It does not mean that the values are correct, as we have seen that packet loss has an impact on the GoP estimation, but we cannot secure they are correct.

So to find some statistics on the RQM estimation, we have to assume that we don't know neither the right value of the GoP nor its statistics and only see the impact of the packet loss error. This will give us a lower bound of the errors and the deviation, which we can see in the table 4.

	Packet loss = 0,1%	Packet loss = 1%	Packet loss = 3%	Packet loss = 5%	Packet loss = 10%
Expected	g(I) - 0,1490317	g(I) - 0,0583	g(I) + 0,0971	g(I) + 0,2005	g(I) + 0,326
Mean	g(I) - 0,1512	g(I) - 0,0601	g(I) + 0,0958	g(I) + 0,2061	g(I) + 0,3159
Variance	4,107E-06	4,236E-05	5,299E-05	2,479E-04	2,110E-04
StDeviation	2,027E-03	6,508E-03	7,279E-03	1,574E-02	1,452E-02
Rel. Error	1,521%	3,168%	-1,314%	2,810%	-3,097%

Table 4: RQM statistics

About the statistics we can mention several things; all the means are close to the expected value, which gives a short relative error. Even though, this relative error is larger than the one in the packet loss estimation. So the error is not only propagated but also increased.

Despite this error propagation, the variance and standard deviation are very low, which tells us that the estimation is good, especially for the simulations with low packet loss rate. The variance and deviation increase for higher packet loss rates. This fact matches with what we see in the figures 32 and 33.

At last we investigate the evolution of the VRQM values all along the estimation of a video, to see how the software responses to the packet loss, the variations on the group of picture and the estimations errors. We can test it, for example, with a packet loss rate of 5%, which will generate scattering in the RQM estimation and most likely will also introduce errors in the group of picture estimation. The temporal evolution of the estimation can be seen in the figure 35.

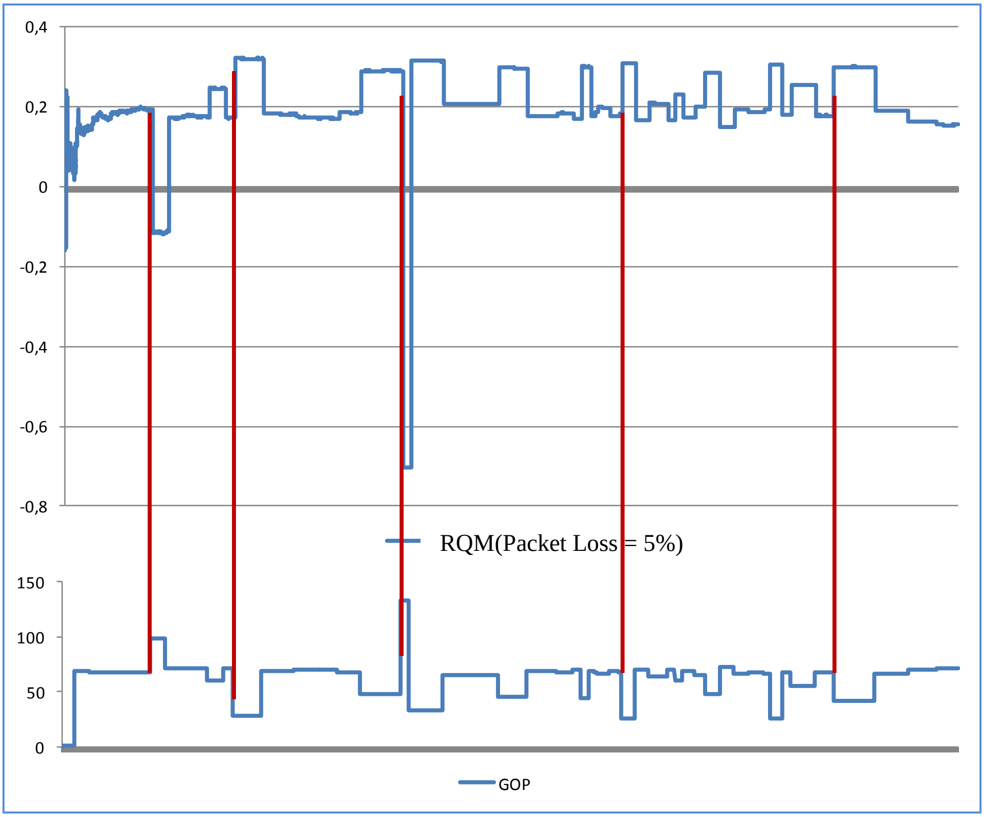

Figure 35: RQM and GOP temporal evolution

With the figure 35 we can observe interesting behaviors. Once we saw the sharp changes, we suspected it could be due to GoP changes, so we added the GoP evolution below and found out that the changes match, so they are due to GoP changes indeed.

Also, the transitory behavior of the packet loss estimation is visible in this simulation as well.

At last, seems like group of picture generates much larger changes than packet loss, which gives small variations.

CHAPTER 7

7 Conclusions and Future Work

7.1 Conclusions

The video quality prediction software has been successfully developed and, taking a look at the simulations results and their statistics, our first conclusion is that: **the software gives a good estimation of the video quality, especially with low bit rates.** We can say so because the simulations results prove that the performance of the estimation is good. Even if there are errors in both packet loss and group of picture estimations, these have generally a small impact on the RQM estimation, according to the variance and deviation values obtained.

Regarding the error and the scattering, we can conclude that **packet loss has a double impact on the RQM estimation;** it does not only generate scattering in the RQM values but also affects the GoP estimation which also generate scattering in RQM and senseless values. However, this impact is not very serious for the packet loss rates that may happen during a real transmission; usually not above 1%.

As we have seen in figure 35, the RQM temporal evolution has almost flat values that change every time the group of picture varies. The packet loss is responsible for the small variations in the value. With that we can say that **packet loss gives a thin adjustment and the GoP large scale variations to the RQM estimation.** And this is due to three facts:

- The packet loss estimation is updated every time a packet is received, while the group of picture is updated every time it varies, every several pictures, which means hundreds of packets.
- The packet loss real value is fixed, so the estimation is always near the fixed value, while the group of picture is not fixed, it can take any value within the margins of the interval.

- Group of picture contribution in the RQM algorithm is larger than packet loss. Most likely because group of picture values are dozens while packet loss is never above 10%.

As we mentioned in the chapter 4, when the software algorithm was explained, the algorithm used to estimate the packet loss rate may output an average rate after a long time. So **in order to be usable in long sessions, some way needs to be found to avoid the estimation of an average value and keep the estimation sensitive to packet loss bursts.** This problem was not detected in the simulations because then the packet loss rate was a fixed value, did not vary.

7.2 Future work

This research is about to be complete but not finished yet.

To start, the problem with the packet loss estimation should be studied. To avoid it to output an average value after a long time simulation some mechanisms could be studied.

One method could be to reset the estimation periodically. It would generate a transitory zone every time it is reset, but it would be always sensitive to critical situation. The period of this reset should be studied; a too short period would never leave the transitory zone, and a too long period would not be sensitive to critical situations.

Another way to keep the sensitivity would be to be aware that after a long time simulating, the estimation takes an averaged value. With that, the software should take into account that the bursts and critical situations will make a smaller variation in the estimation.

Another way in which the research could be continued is to consider a more realistic scenario where other distortions affect the transmission but not only packet loss. It refers to packet reordering, packet corruption, delay or jitter. These distortions are present in the real UDP transmissions, as the packets can go through different paths to get to the receiver. Also, DemoGUI platform would allow us to test the software, as it can also introduce these distortions. With packet reordering the packet loss estimation would be affected, because sequence numbers could be disordered even without loss, packet corruption would make the software read wrong values and delay and jitter would give synchronization problems.

At last, the research could be done again starting from another Full-Reference Model. Even if VQM was chosen because it is the metric with the highest correlation with the Subjective Metric, we have seen that the No-Reference model that has been developed from has little errors. So maybe it would be possible to find a Full-Reference Model with a slightly lower correlation with Subjective Metrics but with which the No-Reference model developed from it would have a better performance. That should be considered because there were several Full-Reference with a correlation coefficient very close to the VQM one.

References

[1] "The top5 video streaming websites", June 2012. http://www.techsupportalert.com/top-5-video-streaming-websites.htm

[2] "YouTube statistics", 2012. http://www.YouTube.com/t/press_statistics/

[3] "Inside YouTube videos", February 2010. http://www.sysomos.com/reports/YouTube/

[4] "IPTV and VOIP subscriber and broadband households worldwide", March 2012. http://www.statista.com/statistics/183622/residential-iptv-and-voip-subscriber-worldwide-in-2007-and-2012/

[5] "IPTV industry statistics", 2009. http://www.iptvmagazine.com/stats.html

[6] M. Rouse, "Quality of Experience", April 2008. http://searchcrm.techtarget.com/definition/Quality-of-Experience

[7] "IP Network Monitoring for Quality of Service Intelligent Support", August 2012. http://projects.celtic-initiative.org/ipnqsis/

[8] F. Kuipers, R. Kooij, D. De Vleeschauwer and K. Brunnström, "Techniques for Measuring Quality of Experience". http://www.nas.ewi.tudelft.nl/publications/2010/WWIC2010.pdf

[9] ITU-T Recommendation J.144rev1. Available : http://www.itu.int/itudoc/itu-t/aap/sg9aap/history/j144/index.html

[10] ITU-T Recommendation J.249. Available : http://www.itu.int/ITU-T/recommendations/rec.aspx?id=7007

[11] M. Vranješ, S. Rimac-Drlje, D. Žagar, "Objective Video Quality Metrics", http://bib.irb.hr/datoteka/320176.Objective_Video_Quality_Metrics.pdf

[12] "Description of Video Qualiti Metric software" http://www.its.bldrdoc.gov/resources/video-quality-research/guides-and-tutorials/description-of-vqm-tools.aspx, Institute for Telecommunication Science.

[13] "A No Reference and Reduced Reference Metric for Detecting Dropped Video Frames", http://www.its.bldrdoc.gov/publications/2493.aspx, Institute for Telecommunications Science

[14] "A Full Reference Method Using Causality Processing for Estimating Variable Video Delays", http://www.its.bldrdoc.gov/publications/10-463.aspx, Institute for Telecommunications Science

[15]"Matlab R2012a Documentation: Statistics Toolbox", http://www.mathworks.se/help/toolbox/stats/nlinfit.html

[16]: “RTP: A Transport Protocol for Real-Time Applications”, http://www.ietf.org/rfc/rfc1889.txt, Internet Engineering Task Force, January 1996.
[17] “HDTV Contribution Codecs” https://tech.ebu.ch/docs/techreports/tr008.pdf, EBU Technical, March 2010.
[18] “Encoding standards differences” http://www.aventuracctv.com/technology/standards.asp, Aventura Technologies, Inc.
[19] “Ffmpeg software official website” http://ffmpeg.org/

List of Figures

List of Tables

Appendix 1

A.1 Implementation

```
#include <stdio.h>
#include <stdlib.h>
#include <unistd.h>
#include <stddef.h>
#include <sys/types.h>
#include <sys/socket.h>
#include <netinet/in.h>
#include <netinet/ip.h>
#include <errno.h>
#include <string.h>
#include <netdb.h>

#define FATAL(msg) \
                        do { \
                        fprintf(stderr,"%s:%d:[%s]: %s\n", __FILE__, __LINE__, msg, strerror(errno)); \
                        exit(-1); \
                        } while(0)

#define MESG "usage: ./server3 #port"
#define BUFSIZE 1400

int main(int argc, char *argv[])
{
        int sock;     /* Original socket descriptor in server */
        int clnt_len;                                          /* Length of client address */

        struct sockaddr_in
                clnt_adr,                /* Internet @ of client & server */
                serv_adr;

        int PORT;                        /* Server Port */

        unsigned char message[BUFSIZE];                        /* message received */

        int seqnum1, seqnum2, seqnum, lastseqnum; /*sequence number (2bytes)*/
        int ssrc1, ssrc2, ssrc3, ssrc4,  /*SinSourceIdentifier (4bytes)*/
        lastssrc1, lastssrc2, lastssrc3, lastssrc4; /*Last SinSourceIdentifier (4bytes)*/
        lastssrc1 = ssrc1;
        lastssrc2 = ssrc2;
        lastssrc3 = ssrc3;
        lastssrc4 = ssrc4;

        float packetsrcv=0;
        float packetslost=0;
        float packetloss=0;

        int npictures=0;
        int GoP=0;
        unsigned int nalutype;
        unsigned int funalutype;
        unsigned int endbit;

        float vqm=0;

/*                  UDP CONFIGURATION                              */

        if (argc!=2) FATAL(MESG);

        PORT=atoi(argv[1]);             /* converts port to a number */

        /* Open socket */

        if( (sock = socket(AF_INET, SOCK_DGRAM, 0) )<0)
                FATAL("socket");

/* Set server address */

        serv_adr.sin_family = AF_INET;
        inet_aton("10.0.0.2", &(serv_adr.sin_addr));
        serv_adr.sin_port = htons(PORT);
         if ( bind(sock, (struct sockaddr *) &serv_adr,sizeof(serv_adr)) < 0)
                FATAL("bind");

/* Do FOREVER */
        int i, size;
        for(i=0;;i++) {

                clnt_len = sizeof(clnt_adr);
                //fprintf(stdout, "ready to receive\n");

/* Receive video packet */

        if((size=recvfrom(sock, message, BUFSIZE,0,(struct sockaddr*) &clnt_adr, &clnt_len)) <= 0 )
                FATAL("recvfrom");


/*                  PACKET LOSS CALCULATION                             */


                packetsrcv++;
                seqnum1 = message[2];
                seqnum2 = message[3];
                seqnum = seqnum1*256 + seqnum2;
//                      fprintf(stdout, "%d     %d      %d\n\n", seqnum, size, strlen(message));
                ssrc1 = message[8];
                ssrc2 = message[9];
                ssrc3 = message[10];
                ssrc4 = message[11];

/*Initialize de SSRCs*/
                if(i==0){
                        lastssrc1 = ssrc1;
                        lastssrc2 = ssrc2;
                        lastssrc3 = ssrc3;
                        lastssrc4 = ssrc4;
                        lastseqnum = seqnum;
                }
/*Packet Loss calculation */
                if((ssrc1 == lastssrc1) && (ssrc2 == lastssrc2) && (ssrc3 == lastssrc3) && (ssrc4 ==
lastssrc4))
                {
                        if(seqnum > lastseqnum)
                        {
                                 if(seqnum > (lastseqnum+1))
                                {
                                        packetslost += (seqnum - 1) - lastseqnum;
                                }
                        }else {fprintf(stdout, !!!!!!!!!!!!!!!!!!!”);} //that was to see when a packet was lost

                        lastssrc1 = ssrc1;
                        lastssrc2 = ssrc2;
                        lastssrc3 = ssrc3;
                        lastssrc4 = ssrc4;
                }

                else { fprintf(stdout, OOOOOO"); //that was to see if the ssrc changed
                        exit(-1);
                }
                packetloss = packetslost / (packetsrcv + packetslost) * 100;

                //fprintf(stdout, "%f\n", packetloss);

                lastseqnum = seqnum;


/*                      GoP CALCULATION                         */

                nalutype = message[12];
                nalutype<<=27;
                nalutype>>=27;
                //fprintf(stdout, "%d\n", strlen(message));
                if(nalutype==28)    //FU Nal Unit Type
                {
                        funalutype = message[13];
```

```
            endbit = funalutype;
            endbit <<= 25;
            endbit >>= 31;
            funalutype <<= 27;
            funalutype >>= 27;

            if(endbit == 1)
            {
                npictures++;
                if(funalutype == 5)
                {
                    GoP = npictures;
                    npictures = 0;
                    //fprintf(stdout, "IDR Picture!!!!!\n");
                }
            }
        } else

        if(nalutype==1){ npictures++;}          //Single NAL unit with NO-IDR Frame
        if(nalutype==5)                         //Single NAL unit with IDR frame
        {
            GoP = npictures;
            npictures = 0;
        }

        //fprintf(stdout, "Pictures received: %d\n", npictures);
        fprintf(stdout, "Packet Loss: %f\t\tGroup of Pictures: %d\n", packetloss, GoP);
        //fprintf(stdout, "Group of Pictures: %d\t", GoP);
        fprintf(stdout, "\t\tVQM = %f\n", vqm);

        vqm= -0.16 - 0.0001*GoP*GoP + 0.0064*GoP +
        0.0003*packetloss*packetloss*packetloss - 0.0092*packetloss*packetloss +
        0.1106*packetloss;

        /* Empty message buffer */
        memset(message,0,BUFSIZE);


    }       /* end of FOREVER */

    close(sock);

}
```