Master's Thesis

Quality of experience for adaptive streaming using HTTP Dynamic Streaming

Stefan Johansson
Daniel Norén

Department of Electrical and Information Technology,
Faculty of Engineering, LTH, Lund University, March 2015.

Quality of experience for adaptive streaming using HTTP Dynamic Streaming

Stefan Johansson dt08sj7@student.lth.se
Daniel Norén ic08dn9@student.lth.se

Department of Electrical and Information Technology
Lund University

Advisor: Maria Kihl, EIT.

March 8, 2015

Printed in Sweden
E-huset, Lund, 2015

Abstract

This thesis investigates how to establish the relationship between OSI layer 7 parameters of video streaming and the QoE of the user, and to evaluate which methods are most fitting for the estimation of QoE. The project is made in co-operation with LTH and Acreo, and is a part of the Next generation over-the-top multimedia services (NOTTS) [7]and the Eco system for Future Media Distribution (EFRAIM) project[1].

The underlying techniques, which form the environment of our research of estimating the QoE, is adaptive bitrate streaming over TCP. The purpose is to investigate how a service, that provides a user with the means to benchmark the received quality of the Over the top (OTT) streaming service, can be built and distributed. Today there exists no such service that takes the viewers subjective opinion into consideration. There have been extensive research on some connected fields and issues but none with a unified solution to streaming adaptive bitrate video over TCP with its particular behavior and effect caused on the streamed video.

In this report we evaluated two different methods of prediction of QoE, Pause Intensity based on the number of pauses and their length during playback, and a Linear bitrate model based on the average bitrate quality and its standard deviation. We also made a small user test with our streaming client software to evaluate the two methods to decide which one is the most beneficial to use. The test showed that although one of the most irritating playback deficiencies is when pauses occur, the linear bitrate model delivered the most accurate predictions.

Keywords: HTTP/TCP streaming, Adaptive steaming, HDS, no reference method, PI, QoE, Linear Bitrate.

Acknowledgements

These are the persons who we want to thank for having been involved and helped us during our work with this thesis.

- Supervisor: **Maria Kihl** for her commitment and assistance during the project
- **Jens Andersson** for always take time to discuss and help us with various types of technical problems
- **Kjell Brunnström** at Acreo AB for his support and suggestion throughout the project
- **Jörgen Gustafsson** for his support and expertise in QoE.

Stefan Johansson and Daniel Norén Lund, March 8, 2015

Table of Contents

List of Figures

List of Tables

Chapter 1

Introduction

Already around 1990 the topic Streaming was introduced and used as another description for Video on Demand. During the next decade from 1990s to the early 2000s the network bandwidth became greater, the access to Internet increase and the protocols TCP/IP and HTTP became standard. [8] These are technical factors that has lead to a rapid growth of multimedia streaming over the Internet over the last years and received tremendous attention from academia and industry.

The robustness of the Hypertext Transfer Protocol (HTTP) together with TCP has made it one of the most important protocols for multimedia streaming. Although other protocols exist such as Real Time Streaming Protocol (RTSP) which is a best-effort approach built for streaming, high capacity Internet reduces the downside of using a connected protocol. In our research we will focus on HTTP/TCP multimedia streaming and ways to objectively measure the user experience from objective metrics. In this project a prototype will be developed as a proof of concept which will show that it is possible in some sense to predict the QoE and the accuracy of the predicted result.

This area has in some ways been handled by articles and research that has been done, but these consist of separate formulas and methods which doesn't put all important aspects together into one consistent solution. We will look into the feasibility with using the measurement methods relevant for our project and what areas has to be further researched and some surrounding areas which has to be taken into consideration and whose effects on the user's experience of the media has to be considered.

We have developed a prototype that processes video streams from the Swedish content provider TV4's streaming service called TV4 Play[16], which uses the

Akamai Content Distribution Network (CDN).

Chapter 2

Background

This chapter will go into the different techniques, measurement methods and other areas which we will use to construct a prototype software, and also explain what important aspects there are to consider and what could be done more in future development of the concept.

2.1 The routes over the Internet and congestion sources

The path from the server to the client consists of a diversity of networks, different techniques and prerequisites. There are several sources where congestion can happen in the network. At the user side it can be that the device, the user connects to the Internet with, is not capable enough. One reason can be that the computer's hardware is not fast enough which means that the CPU, GPU or the software used in the web browser could act as a bottleneck. In the local network at the user's home there usually is an router handling the access interface out to the Internet and if several user's simultaneously uses the shared bandwidth it can cause impairment for bandwidth critical services.

If the user connects via Wifi there could be networks nearby on the same channel causing collisions reducing the available bandwidth. Even the router's hardware and performance could hamper the speed of the Internet connection. On the Internet the bandwidth between the server and end user fluctuates. The load on the nodes transporting the stream of packages to the user varies and if the amount of traffic is too high it is forced to throw packages away, depending on the protocol used this will cause reduced received quality or retransmission causing reduced bandwidth.

At the server side the cause for the user not delivered a good enough quality of the supplied service could be that the server has a too high load to satisfy the user's needs. If there are several possible servers available one will be, from certain criteria, chosen. The selection can be bad and affect the supplied service. The selection process will be explained more in section 2.3.

The path between the server and client is not one defined path through a defined set of nodes. Each package can travel its own distinct way, and the stream of packages spreads out like a river delta with one origin and in the end converge to one destination. What causes this behavior of the streamed packages is the current load on the network. Network load can be different kinds of load depending on which metrics are set to be used as a basis in the package routing rules. The package routing rules decides which way it's, at the current instant, most beneficial to send the package through. This makes it difficult to diagnose the network to find out the reason for a low bandwidth.

2.2 Adaptive HTTP/TCP Streaming

The definition of HTTP Streaming is a bit unclear. A better description of the concept could be Progressive download, because when the client start downloading the file and enough information is available the decoding will start and after that, play-out of the content. The video file stored on the server is divided into smaller fragments usually in the length of 2-10 seconds play-out time, depending on protocols for adaptive HTTP streaming. The small fragments make it possible to switch quality in the middle of the streaming of the video[13] [9].

To stream multimedia with the protocols HTTP/TCP and adapt after the network conditions change to the available channel throughput a few other basic blocks are necessary, namely: video-compression, application-layer QoS monitor, streaming servers and a video decoder.[18] The architecture of these blocks are illustrated in Figure 2.1.

Streaming Server The streaming server is offering streaming media by providing data corresponding to the client's request. To do this it is necessary that a communicator such as a transport protocol, an operating system and a storage system work together. These subsystem constitutes a streaming server[18] .

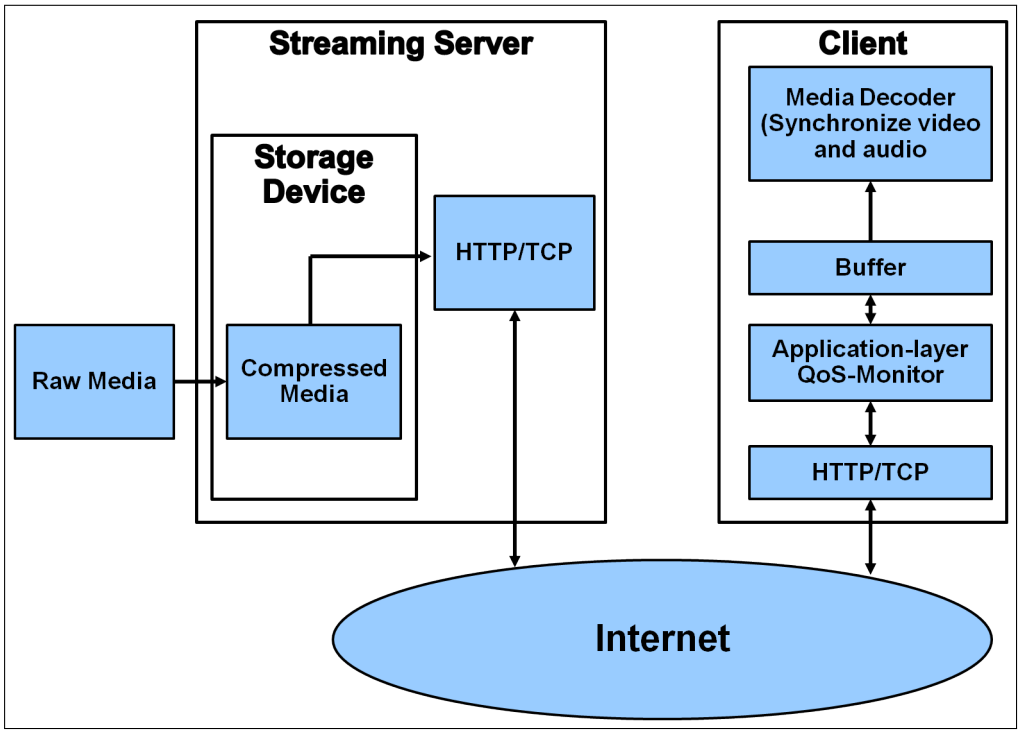

Figure 2.1: Flow chart of streaming from server and client.

Video compression To achieve efficiency during transmission the data must be compressed before it is transmitted from the server. Different encoding rates of the compressed multimedia is then saved in the storage system and each of the compressed video files is divided into the same fragments size which makes it easy for the client to request a different quality of the video stream[18].

QoS monitor The QoS monitor is part of the client and responsible for determining the media bitrate of the file to be downloaded, based on the network condition or the users request for a specific quality. In the rest of the report it will be assumed that the network conditions is monitored and chosen automatically based on current throughput without any specific request made by the user. To cope with altering network conditions the available channel throughput is monitored and based on that parameter the media bitrate will be adapted. Those estimations are done in the client unless in some cases the user decides the media rate. The concept of adaptation after the available channel throughput is a key factor in Adaptive HTTP streaming. [18]

Client The client consists of a similar subsystem such as the streaming server. Both have a communicator and a operating system but instead of a storage device it has a buffer and media decoder[18].

Streaming process As described above, a video file is compressed to different qualities and stored on the storage device. The client will send a request to the server and ask for information about were the video file is stored and which qualities that are available for that video. Information where the video file is stored and available qualities is then sent back from the server. After that the client starts to send requests for the individual small parts of the video clip as in a normal HTTP/TCP transmission. Figure 2.2 describes this process.

There exist four major protocols for adaptive streaming, such as; Apple HTTP Live Streaming (HLS), Microsoft Smooth Streaming (SS), Dynamic Adaptive Streaming over HTTP (DASH) and Adobe HTTP Dynamic Streaming (HDS). All of these protocols work in a similar way, but in this report HTTP Dynamic Streaming will be further investigated because TV4 Play uses it for streaming to desktop computers.

2.2.1 Adobe HTTP Dynamic Streaming

HTTP Dynamic Streaming is a way to stream media from a normal HTTP server (using Adobes HTTP Origin Module for Apache) to a user with a client program showing the media and also at the same time make measurements of the network throughput from the server to control that the maximum available quality of that video is showed. The media file on the server exists encoded into different bitrates, and also every file with a certain bitrate is divided into smaller fragments (.F4F files) for individual downloading, usually 2 seconds.

This enables the client to adapt to its changing conditions and if the client discovers that the download doesn't keep up with the media playout it can get the upcoming part to playout in a lower quality in time, avoiding any pauses in playout for the user. Analogous the client will change to a higher quality of the video if it notices an abundance of bandwidth.

HDS consists of two different ways to deliver content, live streaming and Video on Demand (VOD). When live streaming is used the generation and linking of the generated files is performed in real time and the video is transported using RTMP. In VOD the files available for streaming are pre generated and also the division

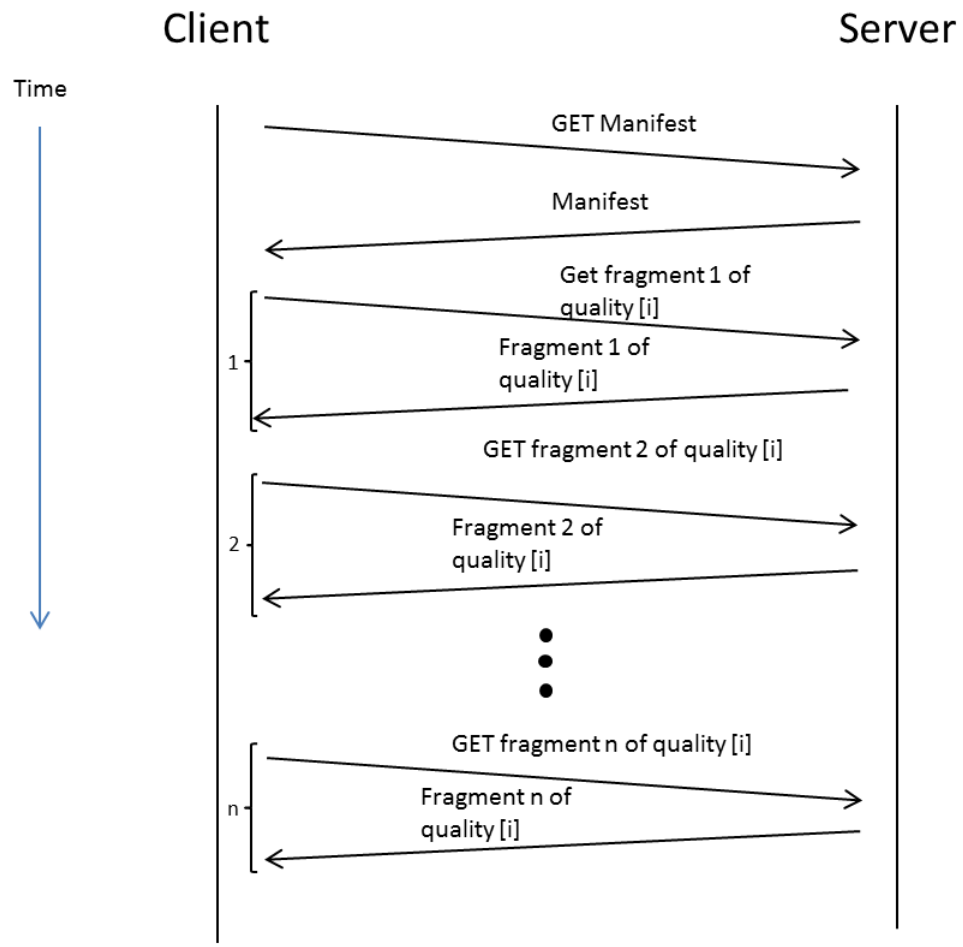

Figure 2.2: This figure shows how adaptive streaming progressively downloads the video, and potentially changes the quality in between get requests.

and linking of the files have been done previously. The files are then streamed with HTTP/TCP. In this report only the VOD part is relevant so the live streaming part will be left out.

The program to create these downloadable fragments is called the File Packager, and it takes the original file encoded in a flash player format. The file to be set available for download is first split into smaller pieces of 2 seconds and then encoded to several versions representing the whole media file but in different bitrates. At the same time as the split happens a file called the manifest is created, it contains all the reference information to all the split pieces and the different

bitrate versions that exists for download.

When a client wants to start stream the media it will first download the manifest file from the server and then decide which bitrate is appropriate to start with. And after that if the client wants to change the quality the client already knows the URLs to the other versions. The manifest contains some more information than just the bitrates, it also contains information about the fragment format, duration of the file, Flash Access license server location for Digital Rights Management (DRM), and other meta data information. The manifest consists of a .F4M file which uses XML formatting[4].

Quality	Fragment index				
300 kbps	1	2	3	4	5
800 kbps	1	2	3	4	5
1500 kb	1	2	3	4	5

Figure 2.3: Layout of the VOD media file on the server

The fragments stored on the server are MP4 fragments in F4F format (ISO/IEC 14496-12:2008)[3].

2.3 Content distribution networks

Companies supplying streaming services wants to provide a service as smoothly as possible. This includes the management and maintenance of the servers delivering the content to the users. The servers are often not the content provider's own servers, instead a content distribution network company is hired by the content provider to supply capable servers.

They provide services such as data storage and bandwidth with which users stream the media from. Content distribution networks are dispersed geographically with several points of presence (PoP) to serve requests as fast as possible[17]. A point of presence is usually a server hall with large capacity. The network load balances the traffic by having a selection mechanism of which PoP should be used, based on metrics such as the server closest to the user (number of hops), round trip time (RTT) and current workload of the different servers etc. The selection algorithm is located in the Dynamic Name servers (DNS) handling a users request

to a CDN host name, and then returns the selected server's IP address to the user.

The major benefit of using a CDN is that the CDN's infrastructure has been optimized for delivery of data to the user as fast as possible. It also adds protection to some extent from Denial of Service (DoS) attacks since the vast capacity and load balancing of the entire network. CDN services is often used for on demand streaming media, social networks and other services where low latency and a broad availability is important both to the user experience and the success of the service.

2.4 Effects of streaming media using the TCP layer

In normal real time streaming services provided over the Internet there are some factors regarding the connection that play a key role in how the media streamed is perceived by the user. It depends on bandwidth/throughput, delay in the network, and how the delay varies called jitter, packet loss, server load and client setup.

In HTTP streaming TCP is used as transport protocol making sure that all packets are received by the client eliminating loss of packets between server and client. The delay and jitter still affects the connection but only in the sense that it reduces the throughput to the client, not the quality received.

2.4.1 QoE

To measure how satisfied a user is of the quality of the streamed media, Quality of Experience (QoE) is used. QoE is according to ITU-T[11]:

"The overall acceptability of an application or service, as perceived subjectively by the end-user"

QoE is a subjective measurement and can only be acquired through subjective testing. The results from a subjective test on a video is a Mean-Opinion-Score (MOS) value which is on a scale from 1 to 5 describes the mean opinion score of all participants that participated in the study.

How users perceive the quality of the streamed media is based some in the underlying Quality of Service properties of the system and network, and also QoS independent properties such as how media rate changes mid-video, or even just how the starting media rate affects the users experience. To objectively estimate the QoE of an user certain parameters/defects of the video are chosen to measure on,

and subjective tests are made with different parameter values to get a correlation of how the values affect the QoE.

2.5 Reference methods

To measure the quality received at the client there exist three types of measurement methods. Full reference, reduced reference and no reference methods corresponding to if the original streamed video is available in any way as a reference.

2.5.1 Full reference

In a full reference method, the received video stream is compared with the original video stream in terms of image quality. This method is often impractical, since verification of the PI requires both the original and the received video to be available at client side and is therefore not applicable to our scenario.

2.5.2 Reduced reference

In a reduced reference method only a small part of the original video is compared to the received one. This could be a small area of the original video compared with the same area on the received video.

2.5.3 No reference

In no reference methods, the properties of the received video stream is measured without any reference to the original video. Instead of comparing our received media with the original clip is properties of the received media correlated to a user perception of the experience. This requires subjective tests being performed on how users commonly reacts to certain streaming conditions, variations in quality and other defects.

2.6 Pause Intensity and Linear bitrate model

Even though there are several different methods they are not all applicable to every problem. In the case of TCP no image quality will be lost, but instead pauses will be introduced caused by buffering and the client not being able to keep up with the media rate with its download rate. This in turn may also cause the client to switch to a lower media rate, also affecting the users perception of quality.

Pause Intensity is a measure where the behaviour of the buffer at the client side is in focus. Pause Intensity measures several aspects such as how long time it takes for the initial buffering when starting the stream, how long it can hold until the buffer is at a state where it has emptied the buffer and needs to pause since no more data is available. These measurements relates to the properties pause frequency and pause mean duration. Both properties are essential of the viewers perception of the quality received. An add-on to this assessment is that it can also matter for the viewer not only how often and long, but where in the video time line they occur[6].

The bitrate of the media and how it varies plays a large part of how the quality of the media is perceived. There are studies showing that even if the bitrate is changed to a higher it can cause the QoE to drop and that a gain of playback quality in the short run can do damage to the QoE . Even what bitrate the media starts playing at can have a great impact on the overall opinion of the media, or what the mean bitrate through the playback is and how much it varies. [19] [12]

2.6.1 Model of Pause Intensity

Pause Intensity(PI) is a no reference method that evaluates the QoE based on pause events. These pause events depend on the amount of data in the buffer. The amount of data in the buffer varies over time, and how the data varies is based on the relationship between the variables λ and η. [6]

- η is the current rate the media arrives into the buffer
- λ is the current rate the media is read from the buffer.

The buffer can have two different behaviours, one if $\eta \geq \lambda$ and another one if $\eta < \lambda$. The buffer flow is illustrated in Figure 2.4. In the first case when more data arrives to the buffer then leaves it, the video will continue to play smoothly without pause, assuming that the buffer is large enough to hold a few seconds of playback. But in the second case when more data leaves the buffer then arrives the number of packet in the buffer will be reduced to q_{min} at some point, this time is denoted t_{v1}.

When this happens the video will pause and wait until the amount of data in the buffer reaches q_{max}. The time when the data has reached q_{max} is denoted t_{max} and playback will then continue. Since more data leaves the buffer then arrives to it, the buffer will probably be reduced to q_{min} again. When the data is reduced

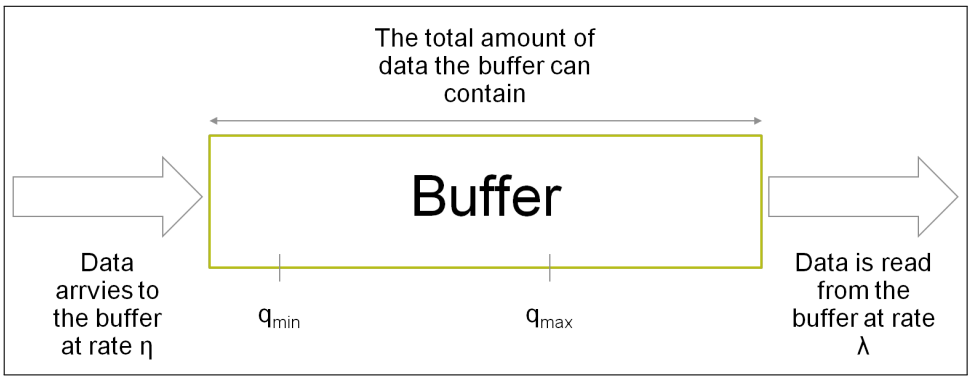

Figure 2.4: Data arrives to the buffer at rate η and leaves the buffer at rate λ. If the amount of data goes below q_{min} the video will pause and wait until there is q_{max} data in the buffer before it continues to play.

to q_{min} for the second time is that time denoted t_{v2} [6].

Below is an explanation how the PI value is calculated based on the buffer behaviour introduced above.

- q_{pause} - is the current amount of data in the buffer when $t_{v1} < t \leq t_{max}$
- q_{play} - is the current amount of data in the buffer when $t_{max} < t \leq t_{v2}$

The amount of buffered data during the pause-play periods can be expressed as:

$$\begin{cases} q_{pause} - q_{min} = \dfrac{q_{max} - q_{min}}{t_{max} - t_{v1}}(t - t_{v1}) & t_{v1} < t \leq t_{max} \\ q_{play} - q_{max} = \dfrac{q_{max} - q_{min}}{t_{max} - t_{v2}}(t - t_{max}) & t_{max} < t \leq t_{v2} \end{cases}$$

During a pause there is no output from the buffer which lead to:

$$\begin{cases} q_{pasue} = q_{min} + \eta(t - t_{v1}) & t_{v1} < t \leq t_{max} \\ q_{play} = q_{max} + (\eta - \lambda)(t - t_{max}) & t_{max} < t \leq t_{v2} \end{cases}$$

- v - is the duration of the pauses
- v' - is the duration of play time
- w - representing the period of a pause-play event

$$\begin{cases} v = t_{max} - t_{v1} = \dfrac{a}{\eta} \\ v' = t_{v2} - t_{max} = \dfrac{a}{\lambda - \eta} \\ w = \dfrac{a\lambda}{\eta(\lambda - \eta)} \\ a = q_{max} - q_{min} = \text{fluctuation area in buffer} \end{cases}$$

The last expression shows how the pause intensity is calculated:

$$\begin{cases} \text{Avg Pause duration} = \bar{v} = \dfrac{a}{\eta} \\ \text{Pause frequency} = \bar{f_v} = \dfrac{1}{w} \\ \text{Pause Intensity} = \bar{v}\bar{f_v} = 1 - \dfrac{\eta}{\lambda} \end{cases}$$

The values PI, Pause frequency $\bar{f_v}$, Avg Pause duration $\bar{v}$ and Max pause duration are mapped to a MOS value. The result of the mapping is shown in Table 2.1.

2.6.2 Linear bitrate model

To predict MOS scores when the original clip isn't available a no reference method must be used. A no reference model uses values measured on the received video to make a prediction. Two no reference models are presented below. Both models are used to predict the MOS score based on the bitrate of the played out video, the values used are called μ_{vq} and σ_{vq}. [12]

- μ_{vq} - is the mean bitrate of the played out video
- σ_{vq} - is the standard deviation of the bitrate from the played out video
- PMOS - is the predicted MOS score

One of the models use a linear approach to predict the bitrate while the other use a power approach. They use the same input values to make the prediction just different coefficients. Since they are no reference models these coefficients be must derived from subjective tests.

$$\left\{ PMOS = k_1\mu_{vq} - k_2\sigma_{vq} + C; (k_1, k_2 > 0) \right.$$

$$\left\{ PMOS' = C' \frac{\mu_{Vq}}{\sigma_{Vq}} \frac{\alpha_1}{\alpha_2}; (\alpha_1, \alpha_2 > 0) \right.$$

[12]

In the models above μ_{vq} is a value from 1 to 5 and σ_{vq} is a value from 0 to 2. Both models work in a similar way regrading the prediction of bitrate changes, a higher average video quality and a lower standard deviation gives a higher MOS value. If the standard deviation is high meaning there are many bitrate changes, studies have shown that it can be damaging for the QoE. [19] The coefficients (k_1, k_2, C) and (α_1, α_2, C') are then balanced for both formulas and both adaptive streaming methods SS and HLS.

The article [12] where the models are proposed has shown that by measuring only the average bitrate there is a 60% correlation with the MOS value. By making use of the models presented above, the article shows that the correlations becomes higher compared with only using the bitrate. The correlation result and the weighted coefficients are presented in tables 2.2 and 2.3.

2.7 Open Source Media Framework

To implement the measurement methods we needed a basis for our software to build upon. The Open Source Media Framework (OSMF) is created by Adobe Systems. OSMF contains a media player called Strobe Media Playback, built upon the OSMF framework. It is written in ActionScript 3.0 which is commonly used for the Adobe flash platform. It supports many video formats out of the box such as the flash format used by HDS. Strobe Media Playback supports all modifications necessary for our implementation, such as listening for streaming and playback events, creation of custom switching rules and an callback interface to Javascript for communication with external sources. [2]

Table 2.1: Pause Intensity values paired with Mean Opinion Score from research

Video ID	PI	Pause Frequency	Avg Pause Duration	Max Pause Duration	MOS
0	0.05	0.06	0.93	1.55	4.35
1	0.05	0.11	0.43	0.64	4.34
2	0.16	0.06	2.80	3.95	4.03
3	0.16	0.11	1.46	2.38	4.19
4	0.23	0.06	4.22	6.07	3.57
5	0.26	0.11	2.31	3.19	3.24
6	0.33	0.06	5.87	9.74	3.42
7	0.33	0.11	3.00	3.67	3.08
8	0.43	0.06	7.72	9.81	2.21
9	0.44	0.11	3.97	8.67	2.41
10	0.51	0.06	9.20	11.66	1.88
11	0.50	0.11	4.47	8.99	1.72
12	0.69	0.06	12.33	18.10	2.05
13	0.64	0.11	5.79	8.38	1.99
14	0.73	0.06	13.16	15.80	1.55
15	0.69	0.11	6.21	8.61	1.43

Table 2.2: Results from the linear model in bitrate changes

PMOS	k_1	k_2	C	Correlation
Apple HLS	1.36	-1.87	1.86	0.90
Microsoft SS	0.91	-1.95	2.06	0.76

Table 2.3: Results from the power model in bitrate changes

PMOS	a_1	a_2	C'	Correlation
Apple HLS	1.58	0.98	0.81	0.85
Microsoft SS	0.92	0.62	1.05	0.81

Chapter 3

Methodology

Here we explain the methods and techniques used to extract the necessary information and the choices we made when these situations occured. Limitations of our work is discussed in section 5.1.

3.1 QoE choices

The user's QoE is based on several factors, where some are interrelated. The factors we have deemed to be the most important ones are; occurring pauses, the duration of the pauses and how the video changes bitrate quality during play-back. In our extensive article research we found no method using all of these factors, but instead we found methods using pause properties and bitrate quality separately. The most disturbing flaw in play-back for the user's QoE is pauses.[12].

The methods we used to measure QoE in our client software are Pause intensity and the Linear bitrate model. In our software we collect the necessary information to use as input to the models.

3.2 Extended Strobe Media Player implementation

The Strobe Media Player (SMP) consists of several different techniques working together. It uses a mix of web HTML pages and JavaScript to be accessible via web browsers. A flash player handling the video stream is embedded in the HTML code. Between the JavaScript and the ActionScript in the flash player there is a programming interface to enable interaction between the webpage and the player. This makes it suitable for distribution by a web server infrastructure.

Since our measurement methods only consider changing circumstances occurring on the Application layer (OSI-7), all information needed for the measurements are available as events in SMP. During playback, the program gather the information and saves it in suitable data structures. When the playback has ended the program do post processing on the data.

One part in SMP that greatly affects the playout performance is the switching rules. Since this is not an instantiation of TV4 Play's flash player, it does not share the same rules involved in the decision making of what quality the flash player should request. At first we wanted to customize the rules in our extended implementation of SMP, but we have not been able to get a hold of TV4 Play's switching rules, and therefore have chosen to use the default ones in SMP. These rules are probably a trade secret since optimization with just the right metrics will result in a more satisfying customer experience.

We created two slightly different versions of our implementation. The first version was used in our user test study where we gather metrics and opinions, used to find the best constants for the Linear bitrate model. The other version is used for demo purposes.

3.2.1 The user test

The user test program collects the following metrics from the video stream:

1. The number of pauses
2. The Duration of the pauses
3. Bitrate quality and bitrate changes

When playback is completed the user is asked to grade the video experience in a HTML form that appears next to the player when the video is done. The metrics gathered by the flash player is written out to a hidden field in the HTML form. So when the form data is submitted to our server, PHP is used to write the result to the database of collected user tests. The procedure of the information we extracted from the user tests is described in section 3.5.

3.2.2 The Demo

The Demo is the same as the user test software with the following differences. It internally calculates the Linear bitrate score with the constants extracted from our user tests. Instead of write out the measurement data to a hidden field in a

HTML form, it only outputs the PI and Linear bitrate score to visible fields on the webpage after playback is completed.

3.2.3 Programming language interfaces

SMP has a method to allow external calls to JavaScript functions and JavaScript calls into SMP using the same namespace as internally used by the respective programming language. This was used to transfer the necessary data between the two main parts of our program.

3.2.4 Internal steps of the demo software

There are 4 stages the software goes through during a prediction of the user's QoE. A visual representation of the four stages can be seen in Figure 3.1.

First it will download the HDS manifest file. The address of the manifest file is already statically stored inside the software. How the link address was determined in the first place is described in 3.2.7. The manifest file is stored on the Akamai CDN and is fetched via a HTML Get request. When it is downloaded the file is read to decide what qualities are available that are then processed by the switching rules to choose a stream to start with.

The second stage is where the player starts streaming the video from one of Akamai's servers. During this stage the player monitors three internal events called buffering, playing and bitrateswitch. When the player is in the buffering stage, the player hasn't got enough data to show any video and is waiting for the player buffer to fill up enough to start playing. When the player buffer has received enough data to start showing the video the player enters the stage playing and starts playing the video on the screen. The bitrateswitch event happens when the internal rules controlling the bitrate quality received decides to change the quality.

These events are used to gather the necessary data for the streamed video, to later calculate both PI and Linear bitrate score. After the second stage when all the video content has been streamed the player processes the collected data and calculates the PI and Linear bitrate score as described in 3.2.5 and 3.2.6.

The last stage of the software is when the two scores has been calculated and shall be made visible to the user. The player then uses an external call interface to a JavaScript function located in the HTML page containing the player. The JavaScript function then interactively print the values of the scores on the HTML

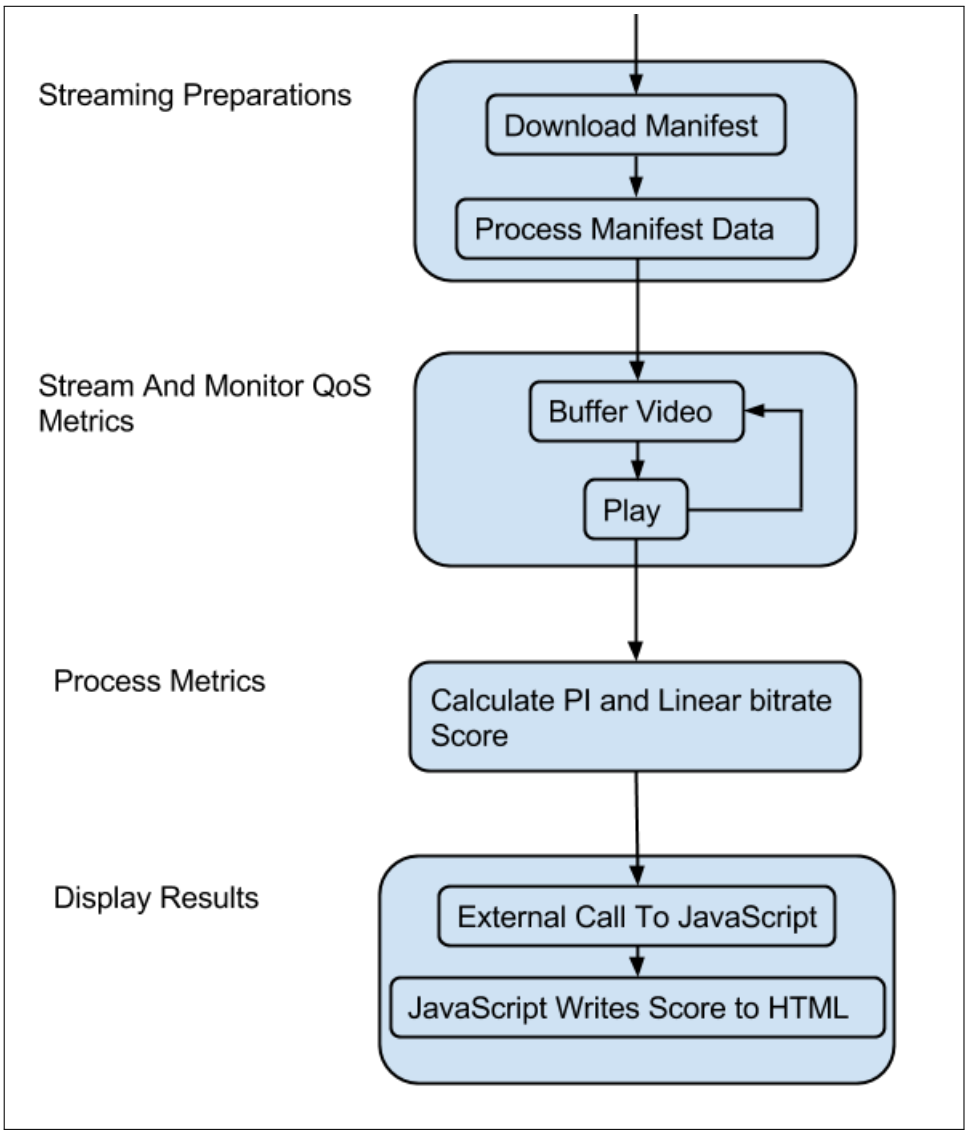

Figure 3.1: The flow of the four stages the software goes through.

page for the user to see.

3.2.5 Pause Intensity implementation

During playback of a video stream, the software monitors two events whose data is used later for the calculation of PI. The events are if the player enters the buffering

state or if the player enters the playing state. When these events occur the timestamp of the event is stored in an array and the type of event in a corresponding index in another array. These two arrays are used to extract the number of pauses and their length in the post processing stage.

The information needed to calculate PI is:

- Number of pauses
- Length of each individual pause
- Length of the video clip

The pause durations are calculated by going through the two arrays to find a buffering event and the following play event and extract the time period in between these two event types. Then the average pause duration is calculated by adding all the pause durations and dividing the total pause duration by the number of pauses. Pause frequency is calculated by dividing the number of pauses with the duration of the video.

The pause intensity is then the product of the average pause duration ($\bar{v}$) and the pause frequency ($\bar{f_v}$). The resulting PI value is then matched to the intervals in Table 2.1. This table is used to match the PI value to a MOS. When a PI value is looked up in Table 2.1, the value is at first found do be in an interval between two PI scores. Since the PI scores are in a range with only the endpoints having MOS scores, it will be matched to the higher PI value. This is made because the PI score has fulfilled the higher PI score bound, but not the lower bound.

Table 2.1 shows that different values of the two factors, average pause duration and pause frequency can cause different MOS although the PI score is the same. We have chosen to match the PI score directly and not evaluate the factors separately to reach the more fine grained prediction. This is made for simplicity of our implementation and that it in overall would result in a small gain.

3.2.6 Bitrate variation implementation

The Linear bitrate score is calculated with the formula in section 2.6.2. The formula requires three constants to be weighted for the specific setup. This is done with our user tests. The most beneficial constant values are explained in section 3.5. This prediction model requires more processing of the collected data than PI.

The information needed to calculate Linear bitrate is:

- Length of all individual intervals with the corresponding bitrate quality
- Timestamp of when a pause occurred
- Length of each individual pause
- Length of the video clip

The event relevant for monitoring, besides what is already used for the PI implementation, is the swapbitrate event. When this event occurs, the new bitrate quality and the timestamp of the event are recorded. After playback the processing of the data takes place. First the time span of each bitrate quality must be determined. Since pauses can occur during playback, and we save timestamps of when an event that changes the bitrate, the duration of the pause must be subtracted from the time interval of a bitrate quality playout.

This procedure is explained in code in 3.3.4. The correct time intervals of each bitrate quality is then divided by the total video length to get the proportion of how long the quality was used. These weights are then multiplied with their corresponding bitrate quality and summed up to get the average bitrate played throughout the video playback. The standard deviation of the bitrate during playback is determined with the code in 3.3.2.

The predicted score is on a scale from 1-5, where 5 is the best available bitrate for the video. This implementation is adapted to TV4 Play's streaming conditions and the bitrates supplied. The Highest bitrate quality from TV4 Play is 2500kbit/s [5]. The highest score will be considered to be when the video is streamed in the highest quality through the video playback. Since 2500kbit/s is not close to the topmost quality available from other sources on the Internet it will not correspond entirely. More about this in section 4.2.

The fact that the bitrate is scaled to a 1-5 scale, and that 5 equals the best available bitrate, with the way the formula was weighted by the user test, there is no guaranties that the lowest available bandwidth is equal to 1. By looking at the formula with the constants, it can be deduced that the lowest bitrate corresponds to a value higher than 1. But the value depends on the available bitrate streams for the video. In the video used to conduct the user test the lowest bitrate corresponds to 1.43 and the maximum standard deviation is equal to 1.94 which is slightly less than the theoretical 2.

3.2.7 Extracting the address for the manifest file

To be able to stream from TV4 Play the manifest file is needed. To get it we set up the package sniffing tool Wireshark to monitor HTML traffic on the Internet interface. Then "http://TV4Play.se" was opened in a web browser where the flash player used to stream the video was embedded. When starting to play, the flash player first requests the manifest file to know all details of the video to be streamed. The collected traffic was then filtered to extract the address of the manifest file. The filter is shown below.

$$http.request.method == GET \&\& http.request.uricontains".f4m"$$

After extracting the address, the manifest file could be downloaded with the wget linux utility. The manifest file contained the available bitrates 302,806,1505 and 2503, duration of the clip together with the relative location of the video files.

```
<?xml version="1.0" encoding="UTF-8"?>
<manifest xmlns="http://ns.adobe.com/f4m/1.0"
                                    xmlns:akamai="uri:akamai.com/f4m/1.0">
  <akamai:version>2.0</akamai:version>
  <akamai:bw>5000</akamai:bw>
  <id>/mp4root/2013-11-05/pid4532540(2480983_,T3MP43,T3MP48,T3MP415,
  T3MP425,).mp4.csmil_0</id>
  <streamType>recorded</streamType>
  <akamai:streamType>vod</akamai:streamType>
  <duration>2650.965</duration>
  <streamBaseTime>0.000</streamBaseTime>
  <bootstrapInfo profile="named" id="bootstrap_0">==</bootstrapInfo>
                                                .
                                                .
                                                .
 <bootstrapInfo profile="named" id="bootstrap_3">==</bootstrapInfo>

  <media bitrate="302" url="0_e6d18240eb52c37a_"
                                    bootstrapInfoId="bootstrap_0">
    <metadata>==</metadata>
  </media>                                      .
                                                .
                                                .
```

```
  <media bitrate="2503" url="3_e6d18240eb52c37a_"
                                bootstrapInfoId="bootstrap_3">
    <metadata>==</metadata>
  </media>
</manifest>
```

3.3 Algorithms

In this subsection we describe the ways we handle the collected data but also how we have obtained this data from our streaming client implementation paired with a description of why we chose to create the algorithms he way we did.

3.3.1 Average bitrate quality

The algorithm getAverageBitrate presented below is used to calculate the average bitrate in order to use the prediction algotithms. In the algorithm the is bitrate mapped to a value between 1 to 5 so it can work properly in the QoE estimation on bitrate changes. In algorithm there is an array called TimeSlotWeights which contains quotas of how long duration [0,1] the quality was used in the play out of the video. The quota corresponds to the duration of the video quality divided by the length of the whole video.

```
function: getAverageBitrate{
 AverageBitrate: int
NumberofBitrateChanges:int
BitrateQualityAtInterval:Array:int - The Bitrate quality
streamed in the interval
TimeSlotWeights:Array:Number - The percentage [0,1] of
the playout duration of this quality to the total
duration of the playout.
HighestPossibleBitrate: int - The highest bitrate of
the video stream


AverageBitrate=0;

for i=0:1: NumberofBitrateChanges - 1

        AverageBitrate += ( 1 + BitrateQualityAtInterval[i]
```

```
        /  HighestPossibleBitrate * 4  ) * TimeSlotWeights[i]

end
}
```

3.3.2 Standard deviation on average bitrate quality

The algorithm getBitrateStandardDeviation returns the standard deviation for the bitrate and it's based on a scaled bitrate value between 1 to 5.

```
function: getBitrateStandardDeviation{
AverageBitrate: int
NumberofBitrateChanges:int
BitrateQualityAtInterval:Array:int - The Bitrate quality streamed
in the interval
TimeSlotWeights:Array:Number - The percentage [0,1] of the playout
duration of this quality to the total duration of the playout.
HighestPossibleBitrate: int - The highest bitrate of the video stream
StandardDeviation: Number


StandardDeviation=0;

for i=0:1: NumberofBitrateChanges -1

        StandardDeviation += (1 + BitrateQualityAtInterval[i]
        / HighestPossibleBitrate * 4)
        - AverageBitrate)^2 * TimeSlotWeights[i]

end
}
```

3.3.3 Prediction of the score based on the bitrate

The algorithm below is called getUserScorePrediction and it is used to predict the users' opinion of the video quality experience. The coefficients k_1,k_2 and C are based on our user test and used in the Linear bitrate prediction formula.

```
function :getUserScorePrediction(){
```

```
AverageBitrate:int
StandardDeviation:Number
PredictionScore:Number


AverageBitrate:Number = AverageBitrate();
StandardDeviation:Number = StandardDeviation();
PredictionScore:Number=0;

k1:Number=0.3;
k2:Number =0.2;
C:Number = 2.4;
PredictionScore = k1 * AverageBitrate -
k2 * StandardDeviation + C;

}
```

3.3.4 Normalizing timestamp offset

During playback the time of when a bitrate change is performed is recorded. The time (milliseconds) the switch occurs, and the new media bitrate is stored when this event takes place. But this is not all data needed to extract for how long a certain bitrate was used. Pauses can occur anytime during playback, making all the following time references offset with the pause duration. To filter out the pauses' effect on the bitrate switch timestamps, the pause duration is removed from from the bitrate switch timestamps.

```
StartTimestampsOfPauses:Array:double
PauseDuration:Array:double
TimestampsOfBitrateSwitch:Array:double
RealTimestampsOfBitrateSwitch:Array:double
RealTimestampsOfBitrateSwitch = TimestampsOfBitrateSwitch

next:double
current:double
pauseTimestamp:double
```

```
for i=StartTimestampsOfPauses.length -1:-1:0

 for k=TimestampsOfBitrateSwitch.length -1:-1:1

  current = RealTimestampOfBitrateSwitch[k]
  next = RealTimestampsOfBitrateSwitch[k -1]
  pauseTimeStamp = StartTimestampsOfPauses[i]

  if (pauseTimeStamp < current && pauseTimeStamp >= next)

   for t=RealTimestampsOfBitrateSwitch.length -1: -1: k -1

    RealTimestampsWhenBitrateSwitchOccured[t] -= PauseDuration[i]

   end

   break;
  end
 end
end
```

The intention of looping over the TimestampsOfBitrateSwitch array from the end, is that if the processing began from the start, all timestamps that would later be compared with the pause-start timestamp would be already shifted. This could result in out of bounds of where the actual event took place and the pauses would be removed from the wrong interval.

3.4 User tests

To get data to evaluate the two different models a user test was set up. A test system running our program on a webserver with PHP as backend was used. We distributed the link to the webserver via a social media website and crowd-sourced our test. The user watched a streamed video of 1.5 minutes while the software recorded any relevant events as explained in 3.2.5 and 3.2.6. When the video playback had ended the user was asked how well the experience was perceived on a 1-5 point scale, with 0.5 increments. The scores' meanings can be viewed in Table 3.1. The results was then appended and saved on the server. In a period of 7 days we collected 53 test samples.

With the test we could extract data to weight in the bitrate model formula

Table 3.1: Scale references

Score	Correspondence
5	Excellent
4	Good
3	Fair
2	Poor
1	Bad

constants for TV4Play's setup. The most common way to weight coefficients in an algorithm is to separate the user test samples into two sets. The first set is called a training set[1] and it is used to calculate which coefficients gives the best results when they are compare against the user results .

The second set is called the verification set and it is used to verify the accuracy of the extracted coefficients. Due to the low number of data samples, the division into two groups would cause the extraction of parameters and validation to fluctuate, depending on how the division was made. This is caused by a too small sample space to get an enough homogeneous set to counter the affect the division had on the results. Therefore we decided to use the whole set for the calculation.

The PI scores based on the streaming conditions was used to get a comprehension of how well the PI model performs when no measurement relating to the varying quality is conducted. With this model we didn't need to divide the test samples into sets, since it could be used directly for verification of the PI model. The user test video clip had the different video streams displayed in 3.2. How the contents of the video, e.g. sports or news, affects the QoE is described in 6.3.

3.4.1 User test setup

The user test we constructed consists of three web pages. The first page contained instructions of what the user would experience during the test and how the rating system worked, as can be seen in Figure 3.3. The second page contained our media

[1] A training set is used to extract parameters to use as a base for extracting as correct coefficients for our formula as possible. We then use the verification set to see how well our estimations of the coefficients from the training set correlates with the user ratings in those tests, to see how well it matches.

Stream #	Video Bit Rate	Frame size	Frame Rate
0	300	1024x576	25
1	800	1024x576	25
2	1500	1024x576	25
3	2500	1024x576	25

Figure 3.2: Dynamic streams used in our user test

player and played our chosen clip with the TV4 Play content from one of Akamai's CDN servers which TV4 uses, as can be seen in Figure 3.4.

After the video clip has ended, a drop down list and a submit button appears right of the media player, where the user can rate his or hers perception of the video. Since we only make measurements on video quality of the playback, we muted the sound during the playback.

The user was sent to the third page after submitting the QoE scores. This was a plain page containing a thank you message for the users contribution.

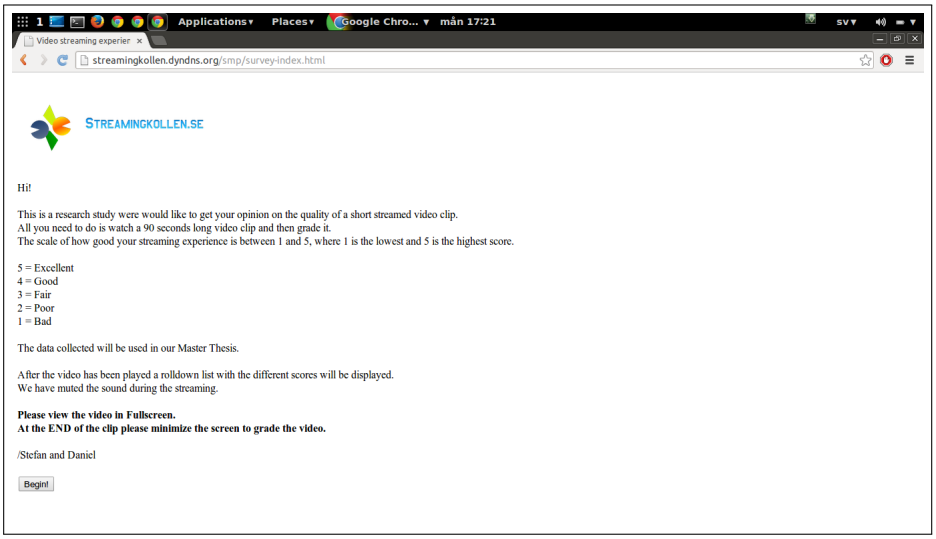

Figure 3.3: This is the introduction web page

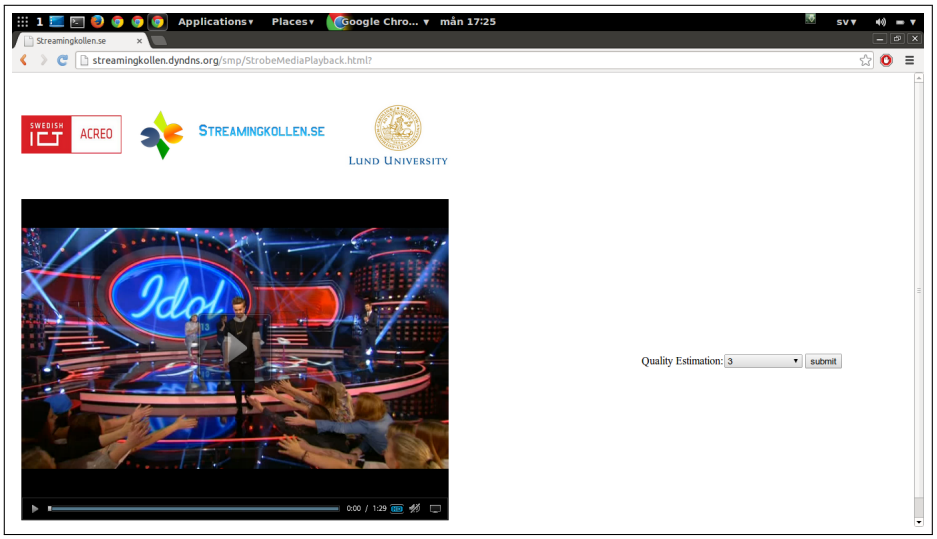

Figure 3.4: This is where the user watches the clip, and then give it a rating, which is submitted by pressing the button

3.5 Processing User Data

To calculate the optimal coefficients and the constant for the Linear bitrate model the algorithm below was used.

It loops through all possible combinations with a resolution of 0.1. It then saves the new triplet if it has a lower Mean Square Error(MSE) than the previous saved triplet.

```
Scores:Array:Number – Contains all the user scoring from
the user tests
PredictedScore:Number – The QoE score predicted by
the Linear bitrate formula
AverageBitrate:Array:int – The average bitrate
quality for the user test with index i
StandardDeviation:Array:Number – The standard
deviation of the bitrate quality in user test
with index i
D:Number – The MSE for the entire set of scores

min=INT_MAX;
mink1=0;
```

```
mink2=0;
minc=0;

// Test all possible combination for k1,k2 and c
// from 0 to 5 with a precision of 1 decimal.
for k1=0:0.1:5
    for k2=0:0.1:5
        for C=0:0.1:5
            D = 0;

            for i=1:1:Scores.length
                //The linear formula to predict QoE.
                PredictedScore = k1 * AverageBitrate[i]
                        - k2 * StandardDeviation[i] + C
                // Using MSE (Mean square error) to
                //determine the best coefficients.
                D = D + ( PredictedScore - Scores[i] )^2
            end

            if (D < min)
                // Store the best coefficients
                min = D;
                mink1 = k1;
                mink2 = k2;
                minc = C;
            end

        end
    end
end
```

To determine the difference between our predictions and the users' scores, i.e. the error, the algorithm below was used.

```
PredictedScores:Array:Number - The QoE scores predicted
by the Linear bitrate formula
AverageBitrate:Array:int - The average bitrate quality
for the user test with index i
StandardDeviation:Array:Number - The standard deviation
```

```
of the bitrate quality in user test with index i
Scores:Array:Number - Contains all the user scoring from
the user tests
E:Array:Number - Contains the error of the prediction
for each user test


for i=1:1:Scores.length

    PredictedScores[i]= mink1 * AverageBitrate[i] -
     mink2 * StandardDeviation[i] + minc

    E[i] = abs(PredictedScores[i] - Scores[i])

end
```

As an addition to the Linear bitrate model where all the coefficients are used and the standard deviation, we also made an analysis by only looking at the average bitrate. This was done in a similar way to find the coefficient k_1.

Chapter 4

Results

In this chapter we present our findings regarding the two QoE prediction models. The models are evaluated based on their ability to accurately predict the user's opinion and their applicability to the QoE benchmark service.

4.1 Results from extraction of coefficients and constant

To be able to analyse the linear bitrate model score, and compare it to the user score, we first calculated the constants needed in the formula. The data come from the result of the user tests made, described in section 3.4. The data collected during the user tests were then processed with the algorithm in section 3.5 to extract the coefficients and the constant needed for the calibration of the formula described in 2.6.2. The constants extracted for the analysis of the linear bitrate model are displayed in Table 4.1.

Table 4.1: Coefficients used in the analysis of Linear bitrate model

Constant name	Value
k_1	0.3
k_2	0.2
C	2.4

The value in Table4.1 corresponds to how well the constants values corresponds with respect to the given opinion score in the user test and the average bitrate. k_1 and k_2 corresponds to the average bitrate and the bitrate variation. C is a constant to correct any offset present from the usage of just k_1 and k_2 in the formula. Since the correlation between the average bitrate and its variation is low at 0.3 and 0.2

they cannot solely be used to get an estimation of the QoE. To compensate for this, the constant C is added to lessen the gap.

The value of the coefficients k_1 and k_2 show that just the bitrate and its variation are not enough for the formula to predict the score accurately. The constant C is used to correct this offset, which is individual for each particular streaming service setup.

4.2 Linear bitrate model

From analysing the user test data, with the formula weighted for TV4Play's setup, we were able to get statistics of the accuracy for the bitrate formula applied on the collected test data. These statistics was used as a measurement of the performance of our two QoE estimation methods.

To find out how close the predicted score is to the users' actual score, we used a histogram of the error values, showing a probability density distribution of the error. The resulting histogram is shown in Figure 4.1.

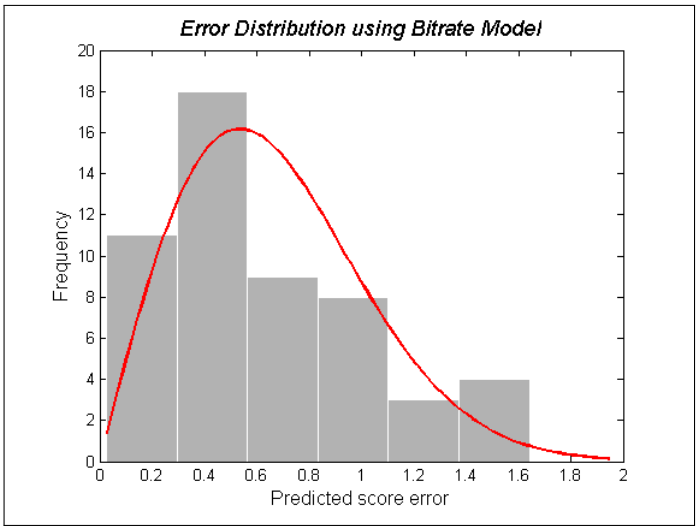

Figure 4.1: Probability density distribution of the error in the linear bitrate model

The distribution of the histogram shows a Rayleigh distribution form where there is a larger probability for an occurrence of a lower Error. The Error is the numerical value for which the linear bitrate prediction formula missed the score, given by the user, on the streamed video with its individual QoS performance.

In our survey with a rather limited number of participants and therefore limited samples we have extracted the 95, 90, 85 and 80 percentiles to see in what interval the error normally occur.

Table 4.2: Percentiles of Error Distribution using Linear Bitrate Model

Percentiles	Score
95	1.47
90	1.35
85	1.06
80	0.89

This shows that the prediction of the QoE using the linear bitrate method gives in 80% of the time an error, equal to or less than 0.89.

Although our purpose with the prediction of QoE is to come in the vicinity of what the users think and our measurements are based on strictly subjective opinions, where videos watched with the same streaming QoS can have varying user scores. Most of the time a satisfactory estimation is reached.

From Figure 4.2 it is obvious that most of the users have registered a score in the 2-4 interval of the 1-5 QoE scale. This makes the balancing of the linear bitrate formula most accurate for estimates in that range. The reason for the ratings being in this range is, probably, an effect of the high capacity Internet of the users and that the video clip used in the test hasn't got full HD resolution as the top quality.

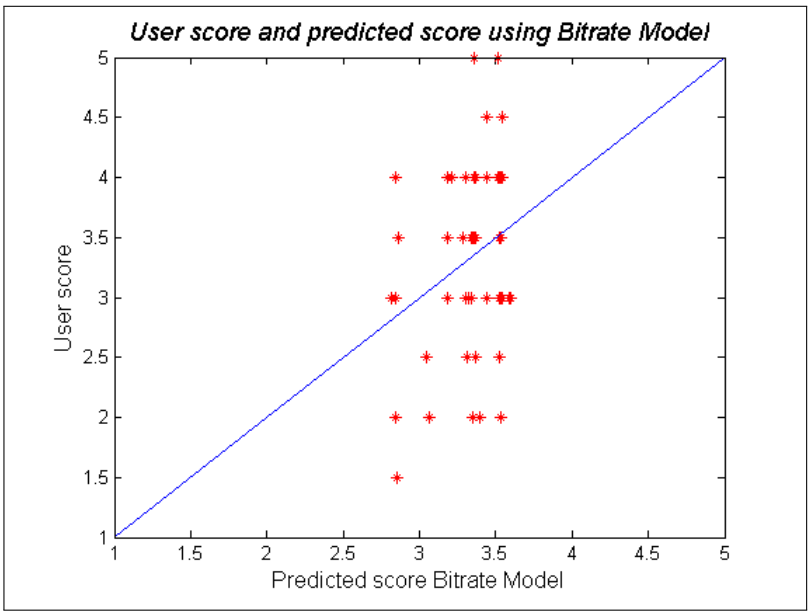

Figure 4.2: This graph shows the clustering of the user score and our predicted score.

4.2.1 Average bitrate model

As a comparison to the Linear bitrate model we investigated how well estimations would be if only the average bitrate was used as an indication. After processing and extraction the coefficient for the average bitrate in the formula, we concluded that 0.82 was the optimal value for k_1. The 80 percentile in Table 4.3 compared to the 80 percentile in Table 4.2 shows that the Linear bitrate model performs better with about 0.3 points.

The offset from the Linear bitrate prediction with just 0.3 points shows that the major indicator in the formula is the average bitrate, and that the variance has a smaller corrective effect to get closer to the user's perceived score. The distribution of the error, using only the average bitrate, is show in Figure 4.4. It shows a logarithmic subsiding distribution with a maximum error at 1.8 compared to Figure 4.1 with a maximum error at 1.6.

Table 4.3: Percentiles of Error Distribution using Bitrate Model

Percentiles	Score
95	1.67
90	1.39
85	1.22
80	1.17

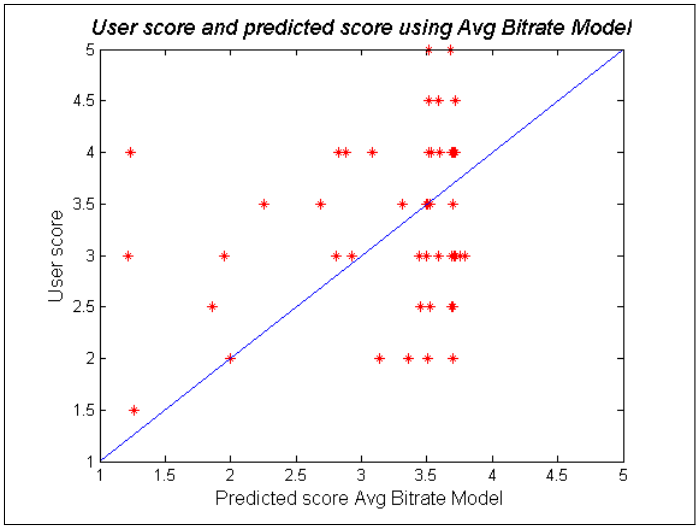

Figure 4.3: Distribution of predicted score with average bitrate index

4.3 Pause Intensity model

In the analysis of the data from the user test, statistics about the accuracy of the PI model were generated. As mentioned earlier the Linear bitrate model performed better than the PI model. The generated statistics resulted in a chart with the user scores on the y-axis and the predicted scores on the x-axis. This chart is presented in Figure 4.5.

The best possible result in PI is 5 and only in six of the cases in the test the predicted score were below 5, however these six does not give any significant result on how the PI model performed. The predicted result was 5 in most of the cases, because no pauses occurred. When adaptive streaming is used a pause can either occur when the rate the media arrives into the buffer is lower than the lowest

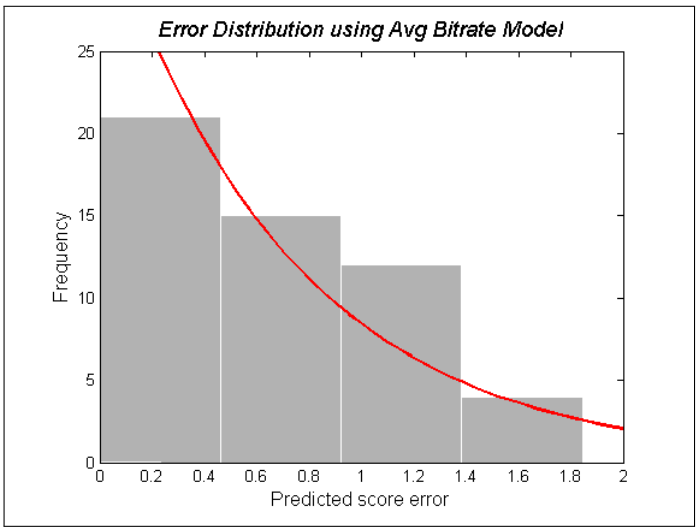

Figure 4.4: Probability density distribution of the average bitrate model

possible playout rate, or if something abruptly happens to the Internet connection and the connection is down for longer time periods than the buffer have stored playback. However, since these cases rarely happened in the user test we conclude that the PI model isn't applicable when adaptive streaming is used and the lowest possible playout rate is achieved. The model however was developed to predict the MOS when only one bit rate is used.

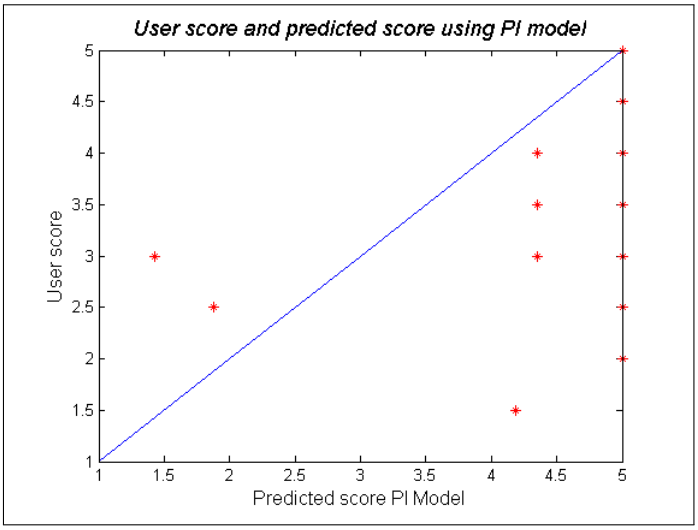

Figure 4.5: Distribution of predicted score using PI estimation vs user rated score.

Chapter 5

Discussion

In this section we present the findings we made during this master thesis and present a proposal of how the streaming benchmarking service could be constructed. We also discuss what areas need further investigation and what effect these investigations will have on the applicability and accuracy of the benchmark service.

5.1 Limitations

In our thesis we had a limited amount of time, which of course introduce several limitations on our disposal of what we can accomplish and focus on. Those limitations are explained in this section.

5.1.1 User test methodology

In our creation of our rather small user test we did not follow an already proven methodology, but reasoned between ourselves to make the user test sufficiently good. The results we present in this thesis is dependant on the verification of our results by a larger test study with a larger test base being collected in a controlled environment. With monitoring of environment variables such as light, computer hardware and the local Internet throughput on the LAN. The way we created our user test may have introduced variance into our sample data which then could have propagated to the results based on that data.

5.1.2 Choice of video content for user test

The clip which we used for the user test was a clip of the TV show "Idol". Thus it introduces two factors that might affect the user's choice of rating the experience. The first is that it is a certain type of video with its own set of communicative

factors being more important, as can be read about in 6.3. The second factor is its inherent property of being "Idol" -some people like it others not, this of course affects a subjective mind that everybody has, and may change the ratings both upwards and downwards. Something more neutral on that front is to be preferred when using the results of the user test to calibrate the Linear Bitrate Model.

5.1.3 Alternative streaming techniques

HDS as we use in our evaluation is just one of several HTTP/TCP streaming techniques and one content provider often support more than one. The reason that we only support one technology is the complexity in the client software used to receive the media stream would be much larger.

To be able to support more techniques the software must be able to handle another protocol and be able to process and display another encoded media content. In our investigation of how the full service could be created we have not put any work into discerning in what context a certain other streaming technique would be chosen by the content provider, their efficiency or any other advantages or disadvantages.

5.1.4 Control of switching rules in the client software

To create a 100% realistic streaming scenario we wanted to implement the same switching rules in our client software as TV4Play uses. This was not possible as we did not get hold of any such rules and proceeded with the standard rules in the library we based the client software on. This of course affects the result since the real flow of the video streaming could not be emulated, unless TV4Play also uses the default switching rules.

5.1.5 Diversion in streaming conditions

When the user test, whose result was used to produce the model for predicting MOS scores, was conducted, the users completed the tests on their own computer at different locations. The test was therefore preformed under different circumstances for each test participant. Since every participant used their own computer the combination of bitrate variation and hardware was unique for each test during the streaming. There were therefore no MOS to compare against, only every participant's opinion of the clip based on their streaming conditions. This makes the result from the test less reliable.

Another limitation was the environment in which the participants completed the test. Factors like people talking or the lightning of the room might have affected the participant's opinion of the clip. Although the use of different test environments might make the test seem more realistic, diversity of external circumstances in tests are never good.

5.1.6 Video quality saturation: Linear bitrate

Things that will be a difficulty when weighting in the formula is that the clip does not contain any High Definition stream as the highest quality. This will cause the scoring of upper sample space to be less frequent, if at all present. And due to the infrastructure of the Internet in Sweden, higher bitrates will be the more common one which will have the effect that also scoring in the lower sample space will be less frequent.

To compensate for this two methods are suggested, one where we construct a monitored experiment where we systematically lower the throughput by disrupting the network traffic by throwing away packages, and then add those "samples" to the other measurements which would somewhat reduce the lack of samples in the lower region.

To compensate for lack of higher scoring we could ourselves supply a High Definition video from a server with a large bandwidth, not using any CDN and controlling the circumstances. This would somewhat compensate the higher scoring region with the disadvantage that we don't have the clip we stream from the CDNs in any higher quality than the one supplied which would force us to choose a similar video, thus probably disrupting the measurement.

The best way to get proper data to calculate the formula's coefficients would be to stream a video with a low minimum bitrate up to a high max bitrate, preferably High Definition with many dynamic streams in between those extremes to reduce the gaps in bitrate when stepping from one video stream to another.

Since this is out of our time limit and would require too much extra work to perform we have limited ourselves to TV4 Play's conditions which will result in a formula weighted, and with the best precision in the mid range of the sample space, where most of the samples occurred.

Another issue with the Linear bitrate model is the fact that it is linear. The model is based on that our perception of video quality is linear but our perception

could be exponentially or logarithmic. If the quality is bad and then it gets a little better it seems likely that it would have a greater impact on our perception compared to if the video quality already is good and gets a little better. The reason that it seems likely is that if you want an apple and then you get one apple you would be happy about it but if you already have five apples and then get one. That one extra apple might not value the same for you if you already have five apples compared to none apples.

5.1.7 Video quality saturation: PI

In the same way, as the limitations imposed by the lack of a high highest quality affects the weighting of the Linear bitrate model, saturation will affect the PI prediction. Even if a really good bandwidth is achieved the top is reached even at medium bandwidth and quality cannot get any better. So the predictions of the PI model will be off in those cases, giving a good rating while the user is unsatisfied and rates the QoE low.

5.1.8 Perception of increased quality relating to QoE

Another issue with the Linear bitrate model is the fact that it is linear. The model is based on that our perception of video quality is linear but it might be the case that it follows another curve form such as exponential or logarithmic. If the quality is bad and then gets a little better it seems likely that it would have a greater impact compared to the case where if the video quality already is good and then gets a little better. Imagine another scenario applying the same kind of thinking to apples instead. If you have no apples and you want an apple, then you would be happy if you got one apple. But if you already have five apples and then get another apple, that one extra apple might not be worth the same for you since you already have five apples compared to when you had none apples.

5.2 Suggested setup of Streamingkollen

The benchmarking service would be most usable by incorporating it into the Web infrastructure. The best way of achieving this is by embedding the software in a webpage. The software needs to support all the major dynamic streaming protocol used by the major VOD providers. The measurement formula best suited for this service in our investigation is the Linear bitrate formula, that depending on the setup of the streaming infrastructure, needs to be calibrated for each setup.

The user selects the service provider to make the test against and then the video start streaming and the QoE parameters required are recorded during the streaming, to later calculate the QoE score and then present it, maybe as suggested in section 5.2.3.

The streaming service providers often support different protocols depending on the device, for example HDS for desktop computers and HLS for mobile devices. This device dependency must therefore be incorporated into the service, with its impacts such as the screen resolution taken into consideration.

5.2.1 Individual baseline for weighting the formulas

The linear bitrate formula has to be weighted to every service provider and streaming technique. Since they each use a different setup with different stepping in media bitrate quality, have varying highest possible media bitrate and use different encoding techniques which greatly effect video quality and size of the media.

The streaming service providers need to have a clip corresponding to their clips normal parameters in regard to quality and scaling of quality representative for their setup.

5.2.2 Software client progressive download rule setup

Progressive downloading of the different parts of the media from the CDN servers is controlled by switching rules in the client software. These rules are up to the service provider to implement or leave as default for the technique or the software used as a base for their customisation.

To emulate the real process of the streaming conditions from a specific streaming service provider these switching rules must be known and implemented in the benchmarking software. These custom rules are a (bandwidth — cost reduction) vs user experience trade off and optimisation for the providers.

5.2.3 Visualisation of QoE prediction score

Since the predicted score has an error margin and that it predicts a subjective view it will have to be taken as an ”in the region” indication. A pure number representation with following decimals from a model based on a continuous function is best visualised graphically. This representation could be in the image of speed meter such as Figure 5.1.

Figure 5.1: Suggested presentation of the QoE score. Image from [14]

5.3 Subjective Testing

This section contains suggestions and guidelines on how to preform subjective tests. If our test had followed these guidelines, it is likely to have been more accurate and reliable.

As mentioned earlier, a test should be done for each streaming provider using the same protocols. All of the participants should view the clip with the same bitrate variations and in the same environment. However, since the bitrate varies with every stream, different scenarios should be viewed by all the participants. Possible streaming scenarios is shown in table 5.2

Possible Streaming Scenarios
Gradually increasing the quality
Rapid increasing the quality
Gradually decreasing the quality
Rapid decreasing the quality
Gradually decreasing and increasing the quality
Rapid decreasing and increasing the quality

Figure 5.2: Possible streaming scenarios during the user test [15]

It is also important that the scenarios are tested on different content in the video clips. Research has shown that the content could have an impact on the QoE [15]. Possible content in the clip could be, Action movie, Drama romance movie, Si Fi movie, News, Documentary, Sport or Music concert.

This means that roughly 36 different clips are necessary to conduct a test. All of these clips should then be viewed by a larger group with different genders and age. The benefits of letting the same people watch the same clips with the same streaming scenarios is that it is possible to get a MOS value for that clip and making it easier to weigh the formula for predicting the score. According to ITU [10], the test should rate on a scale from one to five according to Table 5.1 below.:

Table 5.1: Score references

Score	Correspondence
5	Excellent
4	Good
3	Fair
2	Poor
1	Bad

Also, the data collected by the user test should then be used as a training set calculating the weighted linear prediction formula. The same test should then be conducted with new participants and the result from this test should be used as a validation set to validate the result from the prediction formula.

Chapter 6

Future work

The peripheral areas included in measuring the QoE are many. They also have a large impact on the accuracy of the result and what the result actually means. In this section several topics concerning further research into QoE using dynamic bitrate streaming are introduced and discussed.

6.1 A unified formula

The most important progress to be made is the development of a unified formula incorporating all relevant QoS metrics. The articles we have found during our background research only proposed formulas with a small subset of all metrics, often only 2 or 3 metrics were used.

The Linear bitrate formula combined with our user test reached for 80% of the cases an estimation within 0.89 of the real score, an estimate with an accuracy that can be improved.

6.2 Evolution of the perception of quality

While time progresses the general opinion of what quality is, and how a 5 represent the best quality will be pushed forward with for example when 4k streaming becomes common and thus all the subjective tests made must be redone and updated with the new opinion created by better technology becoming standard.

The content providers must become involved with this type of testing for it to be feasible and accurate. In the way of sharing bitrate switching rules and supplying a video clip representing their standard quality in both bitrate and the dynamic stream stepping between those grades of quality.

6.3 Effects of media type content

The type of video the user wants to see also matters when judging the user's satisfaction. There is a difference on what information in the media streamed that is the most important one. For example in a news broadcast if the video quality would degrade and the news anchor's voice is prioritised and the video quality and not the audio quality is reduced when throughput restricts the data transfer, the user would probably still perceive the most important function of the news cast. Which in this case would be the conveying of the news through the anchors voice.

In other types of media, for example in a sport event such as a football game if the commentators voices would be degraded, it would not effect the user as much as if the video of the game would be grainy and even prevent the user from discerning details or events happening. Which in this case is the primary source of information in the media.

6.4 Geo IP

If the data from running the prediction software is saved in a database paired with the IP the testing was conducted from, a geographical view of the distribution of users and their network QoS condition could be mapped. This could be used to see how certain network operators management of their networks contribute to the streaming experience and would give a possibility for the user to choose its operator based on the performance of the streaming experience they provide. It could also be used to identify bottleneck areas networks or at least give an indication that the performance is an issue.

6.5 Hardware impairments

The devices displaying the video clip, whose streaming properties are benchmarked by our media player, plays a big role in the users' perception of the quality. If the processor doesn't have enough processing power to handle High Definition quality the playout will get laggy. And if the screen resolution has a relatively low resolution in comparison with the video even though a large video bitrate is reached, the user gets an other experience of the video than the streaming metrics shows.

This has to be taken into consideration, and measures should be taken if any of these bottlenecks are impairing the user to actually see the real version of the streamed media. If the computer looses frames in playout the effect for the user

is the same where the picture freezes but internally in the player its the difference of dropping frames when in the playing state or being in the buffering state which has to be monitored and handled accordingly.

Chapter 7

Conclusion

The area of extracting the users QoE from parameters that greatly impact the perceived quality is still in its cradle. The written papers in the area are limited to just viewing one aspect of the streaming's properties such as the bitrate or only looking at the pauses and their duration. The human being is complex in the way how we react on stimuli and then how changes in the stimuli enhances or destroys the feeling of receiving a good service such as VOD.

To move forward a large amount of work is necessary where the streamed video's all aspects are measured, to create one unified formula. There is a clear need of benchmarking VOD to get an indication of how well a streaming service performs and to get a starting point with where to look for the cause of the result.

Since there are many surrounding factors affecting QoE such as in what light conditions the video is being watched, what codec it has been compressed with, what the current opinion is of high definition etc, it is only possible to make an good enough estimation and an estimation model should take them into consideration, so practically researching developing this model is no trivial task. We leave the continuation for the qualified researches around the world investigating QoE everyday.

Bibliography

[1] Acreo. *EFRAIM project*. URL: https://www.acreo.se/efraim (visited on 09/28/2014).

[2] Adobe. *Download OSFM from Sourcec forge*. URL: http://sourceforge.net/projects/osmf.adobe/files/latest/download?source=files (visited on 09/29/2014).

[3] Adobe. *dynamic*. URL: http://download.macromedia.com/f4v/video_file_format_spec_v10_1.pdf (visited on 09/28/2014).

[4] Adobe. *HTTP Dynamic Streaming Specification Version 3.0 FINAL*. URL: wwwimages.adobe.com/www.adobe.com/content/dam/Adobe/en/devnet/hds/pdfs/adobe-hds-specification.pdf (visited on 09/28/2014).

[5] Christer Andersson. *TV4 Play Premium – TV och sport*. 2014. URL: http://www.tv4play.se/premium#premium-heading.

[6] Colin Bailey and Mirghiasaldin Seyedebrahimi et al. "Pause Intensity: A No-Reference Quality Assessment Metric for Video Streaming in TCP Networks". In: *Communications Workshops (ICC), 2012 IEEE International Conference on* (2012), pp. 818 –823.

[7] Celtic-Plus. *Next generation over-the-top multimedia services*. URL: http://www.celtic-initiative.org/Projects/Celtic-Plus-Projects/2012/NOTTS/notts-default.asp (visited on 09/28/2014).

[8] Alexander D. Gelman and Shlomo Halfin et al. “ON BUFFER REQUIREMENTS FOR STORE-AND-FORWARD VIDEO ON DEMAND”. In: *Global Telecommunications Conference, 1991. GLOBECOM ’91. ’Countdown to the New Millennium. Featuring a Mini-Theme on: Personal Communications Services* 2 (1991), pp. 976 – 980.

[9] Ingo Hofmann and Nikolaus Färber et al. “A study of Network Performance with application to Adaptive HTTP Streaming”. In: *Broadband Multimedia Systems and Broadcasting (BMSB), 2011 IEEE International Symposium on* (2011).

[10] ITU. “Subjective video quality assessment methods for multimedia applications”. In: *SERIES P: TELEPHONE TRANSMISSION QUALITY, TELEPHONE INSTALLATIONS, LOCAL LINE NETWORKS* (1999), pp. 1–37.

[11] ITU-T. *ITU-T newslog - QOE CHALLENGE TACKLED IN NEW ITU-T REC*. URL: `http://www.itu.int/ITU-T/newslog/QoE+Challenge+Tackled+In+New+ITUT+Rec.aspx` (visited on 09/29/2014).

[12] Bill Krogfoss and Anshul Agrawal et al. “Analytical method for objective scoring of HTTP Adaptive Streaming (HAS)”. In: *Broadband Multimedia Systems and Broadcasting (BMSB), 2012 IEEE International Symposium on* (2012), pp. 1 –6.

[13] Thorsten Lohmar and Torbjörn Einarsson et al. “Dynamic Adaptive HTTP Streaming of Live Content”. In: *World of Wireless, Mobile and Multimedia Networks (WoWMoM), 2011 IEEE International Symposium on a* (2011), pp. 1–8.

[14] *Speed meter image*. URL: `http://www.clker.com/cliparts/v/c/U/J/W/e/speedmeter-hi.png`.

[15] Samira Tavakoli and Kjell Brunnström. “Subjective Quality Assessment of an Adaptive Video Streaming Model”. In: *Proceedings of SPIE - The International Society for Optical Engineering* (2013), pp. 1–12.

[16] *TV4 Play*. 2014. URL: `http://www.tv4play.se`.

[17] Wikipedia. *Content Distribution Network*. 2014. URL: `http://en.wikipedia.org/wiki/Content_Distribution_Network`.

[18] Dapeng Wu and Yiwei Thomas Hou et al. "Streaming Video over the Internet: Approaches and Directions". In: *Circuits and Systems for Video Technology, IEEE Transactions on* 11.3 (2001), pp. 282 –300.

[19] Liu Yitong and Shen Yun et al. "A Study on Quality of Experience for Adaptive Streaming Service". In: *Communications Workshops (ICC), 2013 IEEE International Conference on* (2013), pp. 682 –686.

Appendices

Appendix A

Abbrivation list

- QoS: Quality of Service
- NOTTS: Next generation over the top multimedia services
- EFRAIM: Eco system for Future Media Distribution
- OTT: Over the top
- HTTP: Hypertext Transfer Protocol
- RTSP: Real Time Streaming Protocol
- CDN: Content Distribution Network
- HLS: HTTP Live Streaming
- SS: Microsoft Smooth Streaming
- DASH: Dynamic Adaptive Streaming over HTTP
- HDS: Adobe HTTP Dynamic Streaming
- VOD: Video on Demand
- DNS: Dynamic Name Server
- DRM: Digital Rights Management
- PoP: points of presence
- RTT: Round Trip Time
- DoS: Denial of Service
- QoE: Quality of Experience
- MOS: Mean Opinion Score
- MSE: Mean Square Error

- PI: Pause Intensity
- OSMF: Open Source Media Framework
- SMP: Strobe Media Player
- HD: High Definition
- ITU: International Telecommunication Union

LUND
UNIVERSITY

Series of Master's theses
Department of Electrical and Information Technology
LU/LTH-EIT 2015-439

http://www.eit.lth.se