NTNU – Trondheim
Norwegian University of
Science and Technology

Evaluating QoS and QoE Dimensions in Adaptive Video Streaming

Julianne M. G. Stensen

Master of Science in Communication Technology
Submission date: June 2012
Supervisor: Poul Einar Heegaard, ITEM
Co-supervisor: Bjørn J. Villa, ITEM

Norwegian University of Science and Technology
Department of Telematics

Problem Description

On-demand content delivery over Internet is growing in popularity, according to Cisco 90 percent of the global consumer traffic by the end of 2015 will be from video traffic. Premium content providers are seeking to create sustainable business models in this area by using innovative streaming solutions. These solutions make the content delivery adapt to the varying network and end system conditions, and by this addressing the objective of achieving a high degree of Quality of Experience (QoE).

Adaptive video streaming is a widely used and demanding example of an "Over-the-Top" service. Technologies for implementing adaptive video streaming require intelligence at the application layer and are developed by different companies such as Microsoft's Smooth Streaming (Silverlight), Adobe's Dynamic Streaming, Apple's HTTP Live Streaming and Move Network's Adaptive Stream. In Norway we see that both TV2 Sumo and also NRK are using the Microsoft solution.

Service providers would like to evaluate the quality of their "Over-the-Top" service. Quality of Service (QoS) captures objective and system-related characteristics while QoE metrics are typically subjective and user-oriented. The focus of this master's thesis will be on bridging these two quality domains in order to better understand the performance of adaptive video streaming. The relation between QoS and QoE metrics will be studied and examples of quantification will be demonstrated by at least one case study where quality is considered from the service provider's perspective.

Steps to achieve this:

1. Study literature and review state of the art on QoS and QoE quantification and bridging the gap between QoS and QoE.

2. Propose quality metrics and the relationship between them applicable to adaptive video streaming.
3. Describe one or several scenarios defining goals and the point of view.
4. Evaluate the use of the proposed metrics on at least one scenario.

Assignment given: 16th of January 2012
Supervisor: Bjørn J. Villa

Abstract

The focus of this thesis has been on Quality of Service (QoS) and Quality of Experience (QoE) dimensions of adaptive video streaming. By carrying out a literature study reviewing the state of the art on QoS and QoE we have proposed several quality metrics applicable to adaptive video streaming, amongst them: *initial buffering time*, *mean duration of a rebuffering event*, *rebuffering frequency*, *quality transitions* and *bitrate*. Perhaps counterintuitively, other research has found that a higher bitrate does not always lead to a higher degree of QoE. If one look at bitrate in relation to quality transitions it has been found that users could prefer a stable video stream, with fewer quality transitions, at the cost of an overall higher bitrate. We have conducted two case studies to see if this is considered by today's adaptive video streaming technologies. The case studies have been performed by means of measurements on the players of Tv2 Sumo and Comoyo. We have exposed the players to packet loss and observed their behavior by using tools such as Wireshark. Our results indicate that neither player take the cost of quality transitions into account in their rate adaptation logic, the players rather strive for a higher quality level. In both cases we have observed a relatively large number of quality transitions throughout the various sessions. If we were to give any recommendations to the Over-the-Top (OTT) service providers, we would advise them to investigate the effects of quality transitions and consider including a solution for handling potentially negative effects in the rate adaptation logic of the player.

Sammendrag

Fokuset til denne avhandlingen har vært på tjenestekvalitet (QoS) og opplevd kvalitet (QoE) i adaptiv videostreaming. Ved å gjennomføre et litteraturstudie med fokus på QoS- og QoE dimensjoner har vi foreslått flere kvalitetsparametre som gjelder for adaptiv video streaming: *initiell bufringstid*, *gjennomsnittlig varighet ved rebufring* , *rebufringshyppighet*, *kvalitets-transisjoner* og *bitrate*. Annen forskning har funnet at en høyere bitrate ikke nødvendivis fører til høyere QoE. Det har blitt vist at brukere foretrekker en stabil videostrøm med færre kvalitets-transisjoner på bekostning av en generelt høyere bitrate. Vi har gjennomført to case-studier for å se om dette en noe det blir tatt høyde for i dagens teknologier for adaptiv videostreaming. Studiene har blitt gjennomført ved hjelp av målinger på spillerne til Tv2 Sumo og Comoyo. Vi har utsatt spillerne for pakketap og deretter observert deres adferd ved å bruke verktøy som Wireshark. Resultatene fra målingene tyder på at spillerne *ikke* tar høyde for en eventuell kostnad i forbindelse med kvalitets-transisjoner. Det virker som om spillerne heller fokuserer på å oppnå et høyere kvalitetsnivå. Vi har observert et relativt høyt antall kvalitets-transisjoner for de fleste av sesjonene. Hvis vi skulle gi en anbefaling til tjenesteleverandørene ville det være å studere eventuelle negative effekter ved kvalitets-transisjoner og vurdere å implementere en løsning som håndterer disse i spilleren deres.

Preface

This report is to be considered as the final product of my master's thesis at NTNU. The master's thesis has been carried out in the 10^{th} semester of the 5-year programme in Communication Technology with specialization in Networks and Quality of Service. The work has been done at the Faculty of Information Technology, Mathematics and Electrical Engineering under the Department of Telematics. At the department I have been guided by my supervisor Bjørn J. Villa and professor Poul E. Heegaard.

I would like to thank my professor and supervisor for the valuable discussions and help I have received during the semester. I would also like to thank Tv2 and Comoyo for giving me access to their Smooth Streaming services. Last, but not least I wish to thank Nina, Silje, Maja, Line and Hanne for the wonderful moments at the office F-257.

Contents

List of figures

List of tables

List of source code listings

Acronyms

CDN Content Delivery Network

CPU Central Processing Unit

CSV Comma Separated Values

DASH Dynamic Adaptive Streaming over HTTP

EPG Electronic Program Guide

FTP File Transfer Protocol

GOP Group of Pictures

HCI Human-Computer Interaction

HTTP Hypertext Transfer Protocol

IP Internet Protocol

ISO International Organization for Standardization

ISP Internet Service Provider

ITU International Telecommunication Union

ITU-T International Telecommunication Union Telecommunication Standardization Sector

MOS Mean Opinion Score

MPD Media Presentation Description

MPEG Moving Picture Experts Group

MPEG-DASH MPEG Dynamic Adaptive Streaming over HTTP

MP4 MPEG-4 Part 14

MSE Mean Squared Error

OTT Over-the-Top

PL Packet Loss

PSNR Peak Signal-to-Noise-Ratio

RTP Real-time Transport Protocol

RTCP RTP Control Protocol

RTSP Real-time Streaming Protocol

TCP Transmission Control Protocol

UDP User Datagram Protocol

VoD Video on Demand

QL Quality Level

QoE Quality of Experience

QoS Quality of Service

XML Extensible Markup Language

3GPP 3rd Generation Partnership Project

Chapter 1

Introduction

1.1 Motivation

We are moving towards an "everything over Internet Protocol (IP)" world, Cisco estimates that the number of devices connected to IP networks will be twice as high as the global population in 2015 [4]. They also expect that in 2015 90 percent of the global consumer traffic will consist of video traffic. This means that enormous amounts of video data is to be transferred over a "best effort" IP network. Meeting customer expectations and providing service guarantees over such a network can be hard, but it is also an interesting field of research.

With the objective of achieving a high degree of QoE, various OTT service providers in this area use innovative streaming solutions. In these solutions the content delivery adapts to varying network and end system conditions. A good understanding of the QoS and QoE dimensions of adaptive video streaming is important for the service providers to be able to evaluate their service.

1.2 Methodology

As stated in the problem description at least one case study is to be carried out to better understand the performance of adaptive video streaming. This has been done in accordance with the steps from the problem description:

1. Study literature and review state of the art on QoS and QoE quantification and bridging the gap between QoS and QoE.
2. Propose quality metrics and the relationship between them applicable to adaptive video streaming.
3. Describe one or several scenarios defining goals and the point of view.
4. Evaluate the use of the proposed metrics on at least one scenario.

As our problem description has been fairly open we want to elaborate our workflow from step 3. In this thesis two case studies have been conducted based on the outcome of step 2. The case studies have been performed by means of measurements on the video players of Tv2 Sumo and Comoyo to see if the proposed quality metrics from step 2 are considered by today's adaptive video streaming technologies. The work with measurement has been an iterative process based on the following steps:

1. Setting up a lab.
2. Planning the measurements, setting up scenarios and goals.
3. Data collection (performing the measurements).
4. Data processing and presentation
5. Evaluation of the results.

1.3 Outline

We start our thesis with an introduction to video over IP, streaming and the Transmission Control Protocol (TCP) in Chapter 2. In Chapter 3 we go into details on QoS and QoE. A description of the objectives for performing measurements, the lab set-up and measurement scenarios is given in Chapter 4, in addition to a describing how we have processed and presented the data. In Chapter 5 we present the results, before discussing and evaluation them in Chapter 6. In Chapter 7 we conclude the thesis together with our views for future work.

Chapter 2

Background

This chapter starts with giving an introduction to the Internet video domain in Section 2.1. Thereafter in Section 2.2 we will go into more details on video streaming. Section 2.2 starts with a historical approach describing "traditional streaming" in Section 2.2.1 for thereby moving the focus towards Hypertext Transfer Protocol (HTTP)-based streaming and adaptive video streaming in Section 2.2.2. In Section 2.3 we give a brief overview on TCP.

2.1 Video over IP

We are moving towards an "everything over IP" world, Cisco estimates that the number of devices connected to IP networks will be twice as high as the global population in 2015 [4]. They also expect that in 2015 90 percent of the global consumer traffic will consist of video traffic. This means that enormous amounts of video data is to be transferred over a "best effort" IP network. Meeting customer expectations and providing service guarantees over such a network can be hard, but is also an interesting field of research. In this section we provide the reader with an overview of what is known as video over IP.

2.1.1 Overview

When we speak of video over IP it is natural to distinguish between the delivery of video over either an *open* or a *closed* network [6] (see in Figure 2.1 on the following page). By an open network we mean the open,

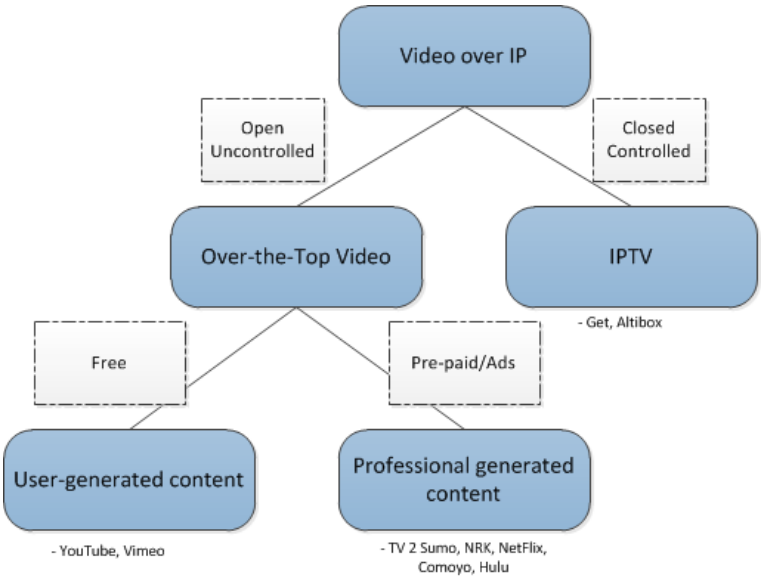

Figure 2.1: Video over IP overview. Figure adapted from [19] (figure 2.2).

uncontrollable Internet – a network which everyone can connect to and provide their own videos or watch other's content. Video delivery over this network is typically done *without* involvement of the Internet Service Provider (ISP) (such as Get, Canal Digital etc.), also known as an OTT service. When speaking of video content delivery over a *closed* network we speak of IPTV, typically managed by ISPs. In Table 2.1 a summary of the differences between IPTV and OTT video is shown. The most important technical difference between OTT video delivery and IPTV is the IPTV's ability to provide service guarantees through their private and managed networks. OTT service providers are on the other hand completely dependent on the underlying best effort network. The interaction with the video player is also somewhat different, where IPTV users typically use a native Electronic Program Guide (EPG) (on the TV) to navigate and decide which content to watch [6]. A OTT video service is typically provided to the end user through a web interface with a plug-in allowing for video streaming. The differences in the interaction plane are although converging as IPTV providers now also let their users interact with their TVs through a EPG in a web-interface, an example of Get's web EPG is shown in Figure 2.2 on the next page.

IPTV represents an evolution of TV rather than a revolution [6]. The user experience is enhanced with means of interaction allowing the user to choose which content to watch (from a pool of content given by the

Over-the-Top Video	IPTV
Public Internet	Controlled network
Generally available on a PC	Generally available on a TV
Best effort basis	Quality of Service guarantees
Streams	Broadcast channels
Downloads	Video-on-demand
Web site	EPG
Free, advertisement, pre-paid	Pre-paid
Open access	Walled garden

Table 2.1: Differences between OTT video and IPTV. The table is adapted from [6] (page 10) where IPTV is denoted service provider-video and OTT video is denoted as broadband-video.

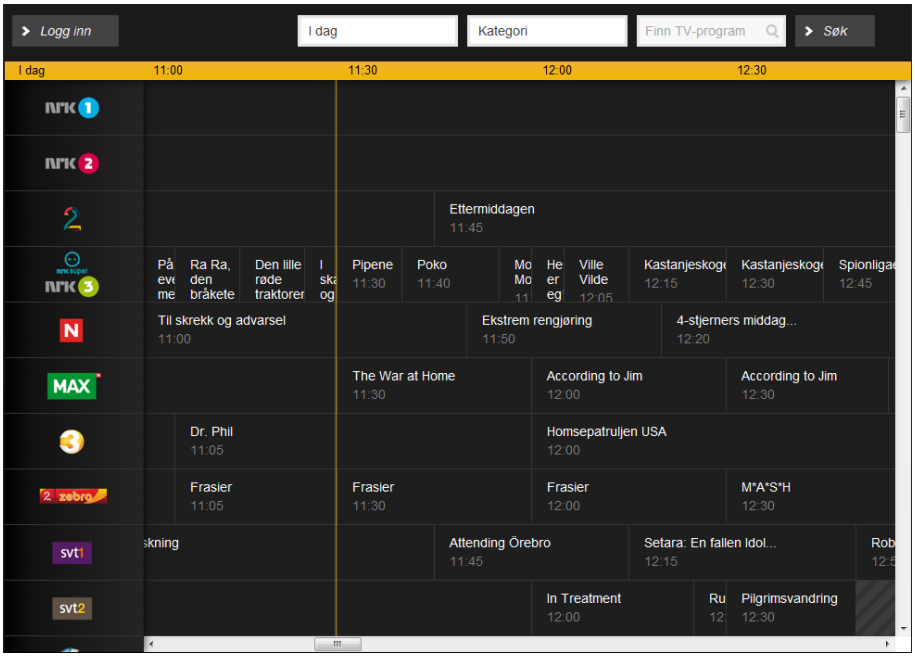

Figure 2.2: Get's EPG from http://www.get.no/underholdning/tv-guide

service provider, a walled garden approach). OTT service providers represent more of a TV revolution where the user is free to explore a large number of free, pre-paid and ad-supported services. OTT video services are thereby an example of disruption in that they provide competition to the traditional TV broadcasting model, although they initially might be technically inferior [6].

2.1.2 Growth and business opportunities

The content offered by the various OTT video service providers can be divided into two categories (as also shown in Figure 2.1 on page 4):

- User-generated content (YouTube, Vimeo, Myspace)
- Professional content (NRK.no, TV2 Sumo, abc.com, NetFlix, Apple TV, Comoyo)

There is an incredible amount of both user-generated and professional video content available, and OTT video together with IPTV represents a new way of watching TV. Users can now choose to watch their favorite show regardless of time and place, and on most devices. The power has shifted from service providers to consumers, which is also one of the reasons for the increased attention to QoE in the last decade. More on QoE is given in Section 3.2. Figure 2.3 on the next page shows the increased amount of scientific work done on QoE since 2000.

Not only is a larger variety of content accessible to end users, but there is also an increasing amount of different devices available. According to a recent study by Cisco [5] the number of mobile-connected devices will exceed the world's population in 2012. This, together with the fact that mobile video traffic exceeded 52 percent of the traffic in the end of 2011, indicates that there is a huge growth potential. On the technical side this growth potential can be stimulated through a standard for HTTP video streaming, more on this is given in Section 2.2.4.1.

Compared to the fixed price economics of broadcasting, where the cost for transmitting a program is the same regardless of how many people are watching, a marginal cost is associated with OTT video delivery. The cost depends on the amount of viewers [6]. This might allow market entrance opportunities for producers of programs attracting a relatively

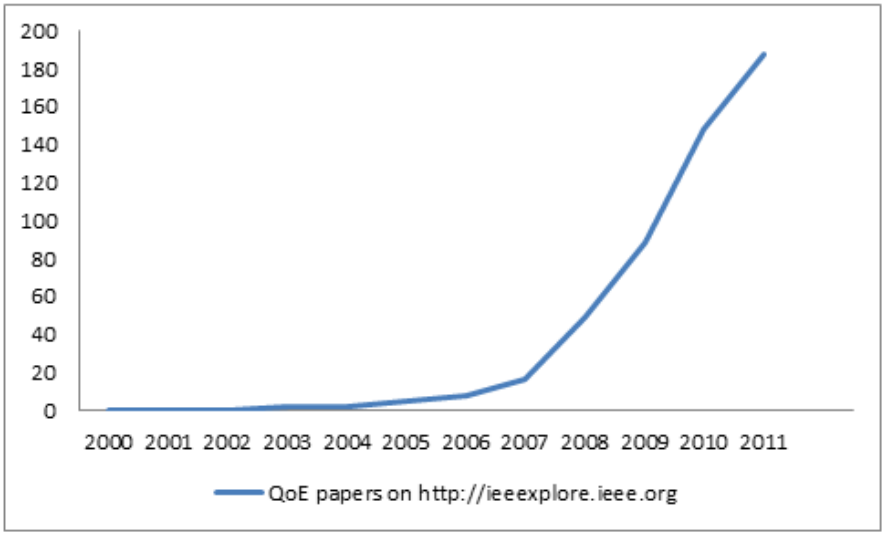

Figure 2.3: Papers on QoE.

small audience. The market is changing and the old business models for broadcasting TV will have to change.

2.2 Video Streaming

Video streaming allows people to access video content over the Internet. It is simply about the transmission of data from a server to one or several clients. The client typically starts the content playback a few seconds after it begins receiving the content from the server, thereby differing from a normal file download where the entire file has to be downloaded before starting the playback. In this eay, the scheme allows for the transmission and rendering of live content.

The most viewed video clip on YouTube has been viewed overwhelmingly 731 240 028 times (as of 29.04.12). Watching video content over the Internet is obviously popular, and one can see more and more actors wanting to be a part of this and thus investing in the deployment of new streaming technology. NRK, Norway's major television broadcaster, announced late April 2012 the launching of their new online TV(http://tv.nrk.no/). There are many providers of such streaming services, among them are TV2 Sumo, Comoyo, Vimeo, NetFlix and Hulu (as depicted in figure Figure 2.4 on the following page), just to name some of them.

Figure 2.4: A screenshot of hulu.com.

Similar for all of them are that their services are run "best effort" over the Internet. To cope with varying network conditions and still be able to provide a good service, several approaches for streaming media has been made. Delivery of media content on the Internet today uses in general three different delivery methods [37]: *traditional streaming*, *progressive download* and *adaptive video streaming*.

2.2.1 Traditional Streaming

Streaming is, as mentioned earlier, about the rendering of an audio/video file while transferring it. Some systems have in addition to this added functionality for user interaction such as pause/resume and temporal jumps within the multimedia file. The Real-time Streaming Protocol (RTSP) is such a protocol allowing users to interact with the media content [16], and is considered by [37] as a good example of a traditional streaming protocol. Traditional streaming protocols are *stateful*, which means that the server keeps and updates information regarding the client's state until the client disconnects. When a connection is established between a client and a server, the server starts sending a steady stream of Real-time Transport Protocol (RTP) packets containing video data. RTSP can also be named a push-based media streaming protocol as it pushes the data towards the client [2]. HTTP is, on the other side, a *stateless* pull-based protocol where the client asks for the desired content. HTTP streaming will be described in more details in the succeeding sections.

Push-based streaming protocols generally utilize RTP as the packet

format for data transmission usually running over User Datagram Protocol (UDP), a protocol without any inherent rate-control mechanisms [2]. This leaves the server the opportunity to push data packets at a bit rate decided upon at the application layer. Adjusting the bit rate to the network might be desirable as packet loss and delay may cause the client's buffer fill rate to be lower than the buffer consumption rate, eventually resulting in buffer underflow and playback interruption.

By monitoring available bandwidth and buffers to adjust the transmission bit rate, stateful streaming protocols, such as RTSP, can provide a smooth playback without pauses and stuttering. The monitoring is usually done on the client side and this information is reported periodically to the server by utilizing the RTP Control Protocol (RTCP) [2]. The server is then in the position of choosing a suitable bit rate for the connection, given that the video is encoded at multiple bit rates.

2.2.2 Towards HTTP-Based Streaming

It is possible to achieve smooth media playback, as mentioned in Section 2.2.1, using the traditional streaming approach. At the same time there are some drawbacks using this streaming scheme. One of the drawbacks is the utilization of UDP for transport protocol as it has a high blocking probability when traversing firewalls and proxies in the network [37][2]. Also, if a traditional streaming protocol such as RTSP is used for streaming it is required to have specialized servers implementing this protocol, which means more costs for the content providers. These are problems not faced if HTTP were used for media delivery. First, HTTP runs over TCP and port 80 and will not have any blocking problems on intermediate network nodes as the Internet is built on HTTP and optimized for HTTP delivery [37]. Second, as most network nodes support HTTP there is no need for specialized servers, thus reducing cost. Also, as HTTP is a stateless protocol, more logic is on the end systems rather than on the servers. This will lead to a more scalable system compared to the case when using a traditional, stateful streaming protocol.

Using TCP for streaming media was unthinkable some years ago. Let me quote a book from 2003 [26]:

> In the case of streaming media, however, a large, and more importantly, consistent bandwidth connection is needed for

> good performance. This is because the streamed packets are played continuously, and the delay involved in retransmission of any lost packets using TCP could affect the performance just as much, or possible more. As a result, UDP which does not use retransmission is used for streaming applications.

This has later on been shown to be wrong; in 2004 a study [33] was conducted, motivated by the knowledge that a significant fraction of commercial streaming traffic was utilizing TCP, to find *under which circumstances can TCP streaming provide satisfactory performance?*. Their outcome was that TCP generally provides good streaming performance when the achievable TCP throughput is roughly twice the media bitrate, with only a few seconds of startup delay. However, such conditions might not be achievable on the public Internet, as network paths might become congested due to competing traffic or if the delivery path does not offer sufficient bandwidth. Streaming adaptively is therefore required for streaming video over TCP [17], a topic we will have a closer look at in Section 2.2.4.

The emerging trend in the streaming media industry is a steady shift away from traditional streaming protocols and back to plain HTTP download [37]. As Zambelli [37] asked: *"Why not adapt media delivery to the Internet instead of trying to adapt the entire Internet to streaming protocols"*. HTTP-based video streaming seems like a good option.

2.2.3 Progressive Download

YouTube is together with Vimeo and MySpace one of the big actors utilizing progressive download for their services [37]. The technique of progressive download is fairly simple; it is nothing more than a normal file download from a HTTP Web server. The player client allows the media to be played back while the file download is still in progress, hence the name *progressive* download. This allows the user to pause the streaming, waiting for the download to complete, allowing a smooth playback when the user decides to press play.

Despite of its popularity, progressive download is not perfect. For instance; it is not very flexible, before starting the download the user has to select a bitrate for the download. This bitrate remains the same through-

out the session, also if there is not enough bandwidth available [2]. This might cause a buffer underflow event leading eventually to an interruption in the video playback. Another disadvantage of this scheme is the case when some user decides to quit watching a video after, for example, ten seconds, after the whole video has already been downloaded to the buffer. This wastes bandwidth. Adaptive video streaming, which we will have a look at in the next section, addresses some of the shortcomings of progressive download.

2.2.4 HTTP-Based Adaptive Video Streaming

Adaptive video streaming is a *hybrid* of progressive download and streaming [2]. It is a concept using the existing HTTP protocol rather than creating a new one. There are several different proprietary solutions based on the same principles for adaptive video streaming. Among them are Microsoft's Smooth Streaming, Apple's HTTP Live Streaming, Adobe's HTTP Dynamic Streaming and Move Networks Adaptive Stream. They are different in many ways, but have in common their use of HTTP and that their media download can be described as a long series of very small progressive downloads, rather than one big progressive download [37].

In an implementation of an adaptive streaming solution the video is cut into many short segments and encoded at the wanted delivery format and rate. The segment length varies from implementation to implementation, but the segments are typically a couple of seconds long. At the video codec level this means that each segment is cut along video Group of Pictures (GOP) boundaries [37], hence there are no dependencies on past and future segments, such that the segments can be decoded independently.

It is the adaptive streaming client who sends a HTTP GET request towards a server to retrieve a particular video segment. This makes the client the managing part. The server is only making the content accessible. In this form adaptive video streaming is similar to progressive download as a video request is done towards a server followed by downloading and rendering of the video segment. The difference lies in the segment size. Short segments enable the client to download only what is necessary.

More importantly, and making this streaming solution more interesting, are its adaptive properties. The adaptive part comes into play when

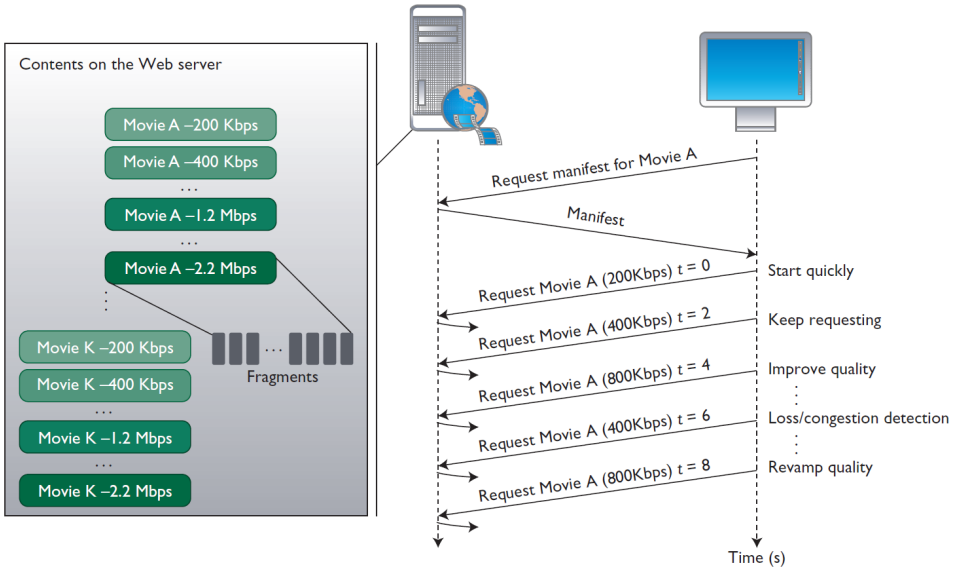

Figure 2.5: Example of client-server communication in adaptive video streaming where the client reacts to varying network conditions [2].

the video is encoded at multiple bitrates [37]. This allows the client to switch, according to some rate-adaptation logic, between different bitrates each time it requests a new video segment. The client strives to achieve the best QoE by displaying the highest achievable quality, providing a fast start-up, fast video seek and reducing skips, freezes and stutters. The rate adaptation and decision logic is based upon monitoring and estimation of related parameters such as [2]:

- Available network resources (i.e. available bandwidth)
- Device capabilities (i.e. display resolution, available CPU)
- Current streaming conditions (i.e. playback buffer size)

Before one can start an adaptive video streaming session, the client has to retrieve information about the video content. Information such as available bitrates and segment duration is of importance so that the client can send GET requests to the server telling which video segment it wants. This information is stored at the server in a *manifest* file. Figure 2.5 shows that this is the first thing the client asks for. The figure also serves as an example on how a client can adjust the requested bitrate to varying network conditions.

Video content available at various bitrates allows customers with a high bandwidth to watch videos at a high quality at the same time as customers with lower bandwidth watch videos at a lower quality. The server simply offers the video encoded at multiple bitrates, while the client is the decision maker deciding which bitrate to pick. The monitoring of CPU usage is useful since it also allows the client to adjust the bitrate according to how much load the CPU can handle. Also the capability to monitor and constrain the playback buffer size is an advantage; it makes it possible to reduce the amount of wasted bandwidth if the customer decides to stop watching the video after ten seconds, a problem with progressive download identified in the previous section.

Another advantage of HTTP-based adaptive streaming is its use of HTTP, reducing the complexity on the server side and there is no need for specialized servers at each node as is if using traditional streaming.

As explained, there are many advantages of using HTTP-based adaptive streaming, but there are also some drawbacks. The lack of multicast support is one of them [2]; in the worst case the server has to send the video as many times as the number of users requesting it. This drawback is often compensated by utilizing Content Delivery Networks (CDNs) for delivering data. In CDNs the data is cached on distributed servers placed strategically around in the network. This reduces the load for the origin server in addition to that the path to the content can be shortened significantly [2].

2.2.4.1 MPEG-DASH

Multimedia streaming over the Internet is still in its infancy compared to its potential market [29]. One reason is that every commercial platform is a closed system with its own proprietary solution. There is no interoperability between these solutions. Apple's HLS, Microsoft's Smooth Streaming and Adobe's Dynamic Streaming are all examples of solutions with their own manifest format, content format and streaming protocol. A HTTP streaming standard would be an instrument for market growth allowing different streaming solutions to work together rather than in parallel. Having a HTTP streaming standard will allow a standard-based client to receive content from a standard-based server.

In April 2009 the Moving Picture Experts Group (MPEG) issued a Call for Proposal for a HTTP streaming standard. 15 full proposals were re-

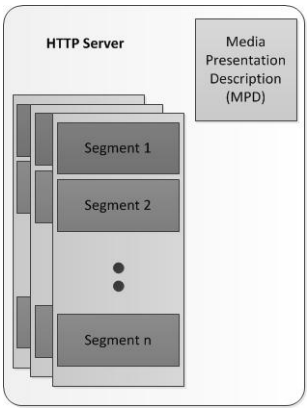

Figure 2.6: Content on HTTP server. Adapted from [29], figure 2.

ceived and in the two following years a standard specification has been under development with participation from many experts and with collaboration from the 3rd Generation Partnership Project (3GPP) [29]. The resulting standard MPEG Dynamic Adaptive Streaming over HTTP (MPEG-DASH), now known as ISO/IEC 23009-1, was frozen technically in August 2011, and ratified before MPEG-98 (the 98th meeting of MPEG) wrapped up in December 2011. MPEG-DASH was finally published by International Organization for Standardization (ISO) in April 2012.

Companies like Netflix, an American provider of on-demand video streaming, sees advantages by utilizing the MPEG-DASH standard:

> "The biggest advantage to us of a standard like MPEG-DASH is that everything can be encoded one way and encapsulated one way, and stored on our CDN servers just once. That's a benefit both in terms of saving our CDN costs from a storage perspective and a benefit because you have greater cache efficiency," said Mark Watson, senior engineer for Netflix[30].

The MPEG-DASH specification provides us with a standardized description of the content stored on a HTTP server. The content which is stored at the server can be divided into two parts (visualized in Figure 2.6):

- Media Presentation Description (MPD)
- Segments

The MPD is a XML-based manifest file describing the available content and its various characteristics, while the segments contain the actual multimedia bitstreams in the form of chunks.

MPEG-DASH is about describing what is on the server. Delivery of the MPD and the media-encoding formats containing the segments, as well as client behavior for fetching, adaptation heuristics, and playing content, are out of MPEG-DASH's scope [29].

2.2.4.2 Microsoft's Smooth Streaming

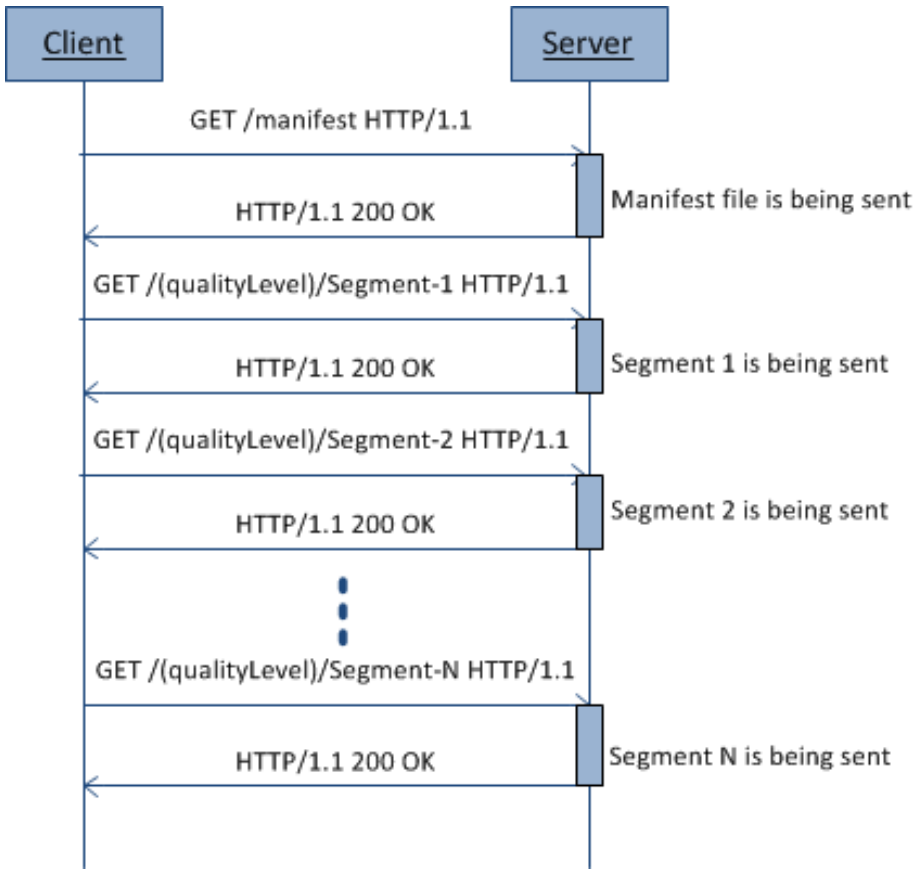

Figure 2.7: Microsoft's Smooth Streaming client-server communication.

As already mentioned, Smooth Streaming is Microsoft's solution for HTTP-based adaptive video streaming. It was in October 2008 that Microsoft first announced that they would feature a new streaming extension called Smooth Streaming. Promotion of this new technology was done initially, together with Akamai, through a showcase web site: SmoothHD.com

[37]. The Smooth Streaming technology is realized through *Silverlight*, a free web-browser plug-in developed by Microsoft [21]. As the further work of this thesis is based on this technology, a more detailed explanation of Microsoft's Smooth Streaming is given here.

The Smooth Streaming technology uses the MPEG-4 Part 14 (MP4) media format (ISO/IEC 14496-12). As described in Section 2.2.4, the video is partitioned into many short segments and encoded at the wanted bitrate. In the Smooth Streaming specification this means that each segment is a MP4-fragment. All fragments are stored together in a contiguous MP4 file, hence there is one MP4 file on the server for each encoded bitrate. The video segments can be accessed at random access thanks to MP4's support for payload fragmentation [37]. The streaming client (Silverlight) is now free to ask for a specific segment encoded at a specific bitrate from the server, which in turn will return the requested segment as a standalone file over the Internet. Figure 2.7 on the previous page shows the overall client-server communication.

Before the retrieval of video segments, a manifest file is requested from the server. As mentioned in Section 2.2.4, this manifest file includes information about the video content, such as available bitrates, segment duration and video start-time. The manifest file is based on the Extensible Markup Language (XML) and in listing 2.1 a simplified example is shown.

```
<?xml version="1.0" encoding="UTF-8"?>
<SmoothStreamingMedia MajorVersion="2" MinorVersion="0"
   Duration="2300000000" TimeScale="10000000">
  <StreamIndex Type = "video" Chunks = "115" QualityLevels
      = "2" MaxWidth = "720" MaxHeight = "480" TimeScale
     ="10000000" Name = "video" Url="QualityLevels({
     bitrate},{CustomAttributes})/Fragments(video={
     start_time}">
     <QualityLevel Index="0" Bitrate="1536000"
        FourCC="WVC1" MaxWidth="720"
        MaxHeight="480" CodecPrivateData = "..." >
      </QualityLevel>
      <QualityLevel Index="5" Bitrate="307200"
         FourCC="WVC1" MaxWidth="720"
         MaxHeight="480" CodecPrivateData = "...">
      </QualityLevel>
       <c n="0" d="20000000" />
```

```
      <c n="1" d="20000000" />
      ... <!-- fragment definitions omitted -->
      <c n="114" d="19680000" />
      <!-- end fragment definitions -->
  </StreamIndex>
</SmoothStreamingMedia>
```

Listing 2.1: Microsoft manifest sample, adapted from [20].

The <StreamIndex> tag in the manifest file specifies the metadata for one type of track (audio, video or text). From the manifest file sample we can for instance see the specification of the URL for the video track, which is in the form of a RESTful URL; clients use this address to retrieve video segments. Video segment requests typically look like this:

```
GET http://video.foo.com/NBA.ism/QualityLevels(400000)
/Fragments(video=6102275114)
```

The value coming after "QualityLevels" is the encoded bitrate (400000). The value is given in bits per second, meaning that this requested video segment is encoded at 400 kbps. The value next to "Fragments" is the fragment start offset (550450000000), this value is expressed in an agreed-upon time unit (can be seen in the manifest sample as the attribute "Timescale") [37]. The recommended value is 10.000.000 which maps to increments of 100 nanoseconds [20]. In the manifest file the duration of each fragment is announced in the tag <c n="0" d = "20000000">, where n is the fragment number and d is the duration (in Timescale increments). The default fragment duration is 2 seconds. With this information the player knows the start offset for the next fragment (previous start offset + duration) and can then send the correct request for the next segment.

The quality level is requested according to which available bitrates that are announced in the manifest file. The announced bitrates can be found after the "QualityLevel"-tag together with the attribute "Bitrate=" (see the manifest sample in listing 2.1).

"There is much more to building a good player than just setting a source URL for the media element"[37]. The server is just making its content accessible, while the client decides how and when to switch bitrate. In the Silverlight framework the clients have a pre-defined way of adapting the stream to the environment, but it is also possible for developers to implement their own version of the rate-adaptation process.

The most challenging part of Smooth Streaming Silverlight development is the heuristics module determining how and when to switch bitrates, but there are also other things to consider such as [37]:

- The size of the download buffer
- What happens if the user has enough bandwidth, but not enough Central Processing Unit (CPU) power for consuming the high bitrates?
- What if the resolution of the best available video stream is larger than the screen resolution?
- What happens if the video is paused or hidden in the background?

2.3 TCP

TCP is a connection-oriented protocol providing reliable transmission of data in addition to mechanisms such as flow- and congestion control. which also is widely used for the transport of video content. In this section we will provide a brief overview of TCP, for more details we refer the reader to [31].

TCP focuses on reliable and accurate delivery rather than timely delivery. If a data packet is lost or corrupted on the way to its destination it will be retransmitted, which in turn might introduce some delay as the packets must be received in order. In TCP the sender adjusts the sending rate to fit the minimum of what the network and receiver can manage. The receiver announces to the sender its "receiver window" to inform about its buffer size. On the sender side a "congestion window" is maintained to adjust the sending rate to the available bandwidth in the network.

One can divide a TCP flow into two phases: a connection phase and a transfer phase. The connection phase consists of a three way handshake for setting up a connection between the sender and receiver. After the connection set-up the TCP flow will enter the transfer phase. The transfer phase consists mainly of a slow start and a congestion avoidance period.

2.3.1 Slow Start and Congestion Avoidance

Figure 2.8 shows a typical transfer phase. After the connection set-up the congestion window is set to be the maximum size of one segment. A threshold value is also set (for example 64 kB). The sender starts sending the amount of data as allowed by the congestion window. The window is doubled when ACK is received. The congestion window will continue to increase exponentially until a timeout occurs or the receiver's window is reached. Although it is an exponential growth period, this period is called "slow start".

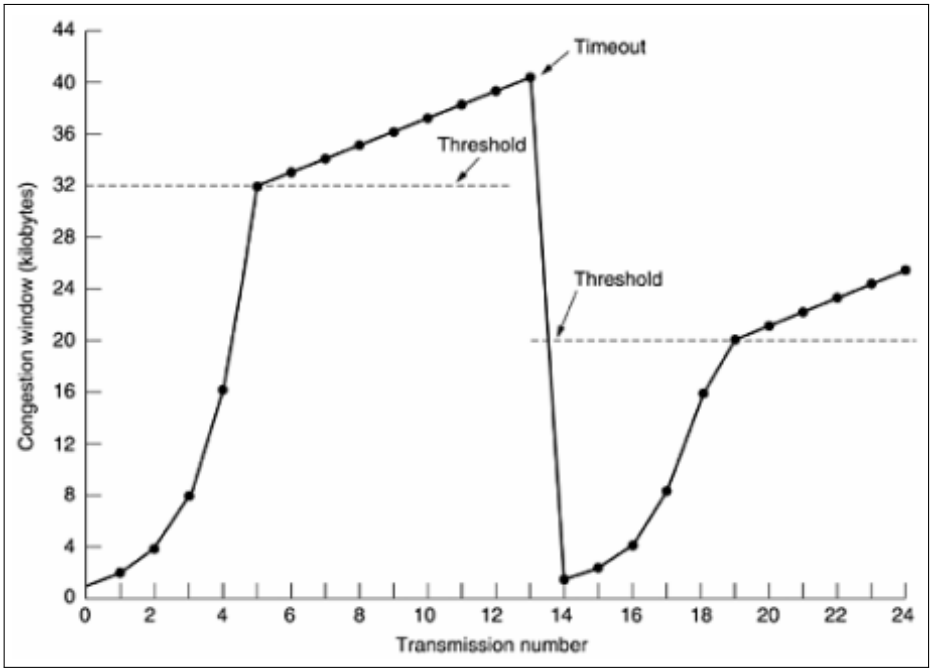

Figure 2.8: The transfer period of TCP. Figure from [31].

When reaching the threshold value, the slow start period is over and the congestion avoidance period takes over. This period is not as agressive as slow start, but the congestion window will keep increasing until a timeout is received. The threshold is then set to half of the current congestion window, and the TCP flow will enter slow start again.

This is the typical life of a TCP flow, but there are many versions of TCP today, and the congestion window evolution might not be as "straight forward" as in Figure 2.8. Some exisiting versions are Tahoe, Reno, Vegas, Compound, CUBIC. As Dynamic Adaptive Streaming over HTTP (DASH) systems utilize TCP for transport it is good to be aware

of that TCP's possible influence the achieved throughput as it is also – as DASH is on the application layer – adaptive.

Chapter 3

Quality

This chapter starts with providing an overview of the concept of quality and motivation for research on QoS- and QoE-dimensions in the Internet domain in Section 3.1. Thereby we look at the increased interest of QoE in Section 3.2. In Section 3.3 a top-down approach on video quality is provided, firstly giving an introduction to video quality influence factors for further on looking at quality assessment methods in Section 3.3.1. The focus will gradually move towards quality assessment for adaptive video streaming. Section 3.4 we discuss why and to who quality measurement is important. In the last section, Section 3.5, a proposal of metrics applicable to adaptive video streaming is given.

3.1 Internet Quality

Network parameters such as packet loss, delay, bandwidth and jitter (packet delay variations) affect the performance of applications running over a network [9]. The tolerance or degree of sensitivity to each of these parameters varies from application to application. Too much delay can, for instance, have fatal consequences in an air plane control system while the same delay in the coffee machine on a workplace does not cause any more than some irritation among the employees. Guaranteeing for the delivery and reliability of a service is a requirement for critical applications such as the air plane control system .

In traditional circuit-switched networks, like the telephone network, service guarantees for a connection can be made and measured in terms

of setup delays, voice quality etc . In packet-switched networks, such as the Internet, this is more difficult as the networks were not designed to deliver per-connection service guarantees [9]. The current Internet architecture is based on the best-effort service model where packets might flow towards the destination through various paths. The different data packets are then likely to experience different amounts of delay, loss and jitter along the path. This has been no problem for traditional applications such as e-mail, File Transfer Protocol (FTP), HTTP and telnet as they are not delay sensitive in addition to their utilization of TCP which ensures a reliable data delivery. Network fluctuations (described through loss, delay etc.) are although likely to have a negative impact on many of today's large variety of applications, amongst them OTT video services. Few service guarantees can be given for applications using the best-effort network infrastructure; this is in contrast to the fact that many of these applications are core components of businesses [14]. Research on QoS and QoE is one way to address this issue.

3.2 From QoS to QoE

The already mentioned network parameters are typical QoS parameters affecting the performance of networked applications. Specification of such QoS parameters usually depend on the context of the involved applications, but the following parameters are considered the basic form of QoS as other forms can be mapped to them [9]:

- **Throughput** refers to the amount of data that can be transferred from a source to a destination per time unit (bits/second).
- **Goodput** is related to throughput, but only consideres the packet payload.
- **Delay** is the time interval between data departure and arrival.
- **Packet jitter** refers to packet delay variation.
- **Loss** is data packets not reaching their destination.

For a long time QoS has been a dominating research topic in the communication networks' domain. QoS was defined by [38] as "providing

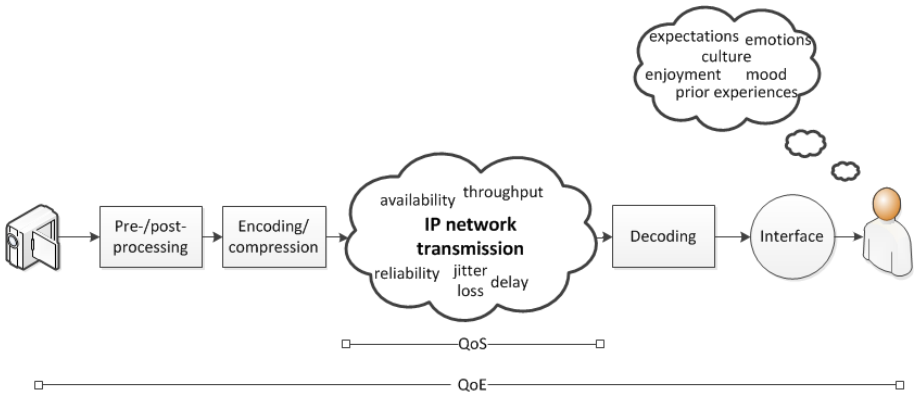

Figure 3.1: QoS and QoE overview for video streaming, figure adapted from [19](figure 2.6) and [8] (figure 1).

service differentiation and performance assurance for Internet applications". This was supposed to be realized through QoS architectures like *Integrated Services* and *Differentiated Services* intending to "pave the way" for high-quality real-time services like video streaming and Voice-over-IP [10]. The notion of end-to-end QoS was originally (for more than 10 years ago), according to the International Telecommunication Union Telecommunication Standardization Sector (ITU-T), aiming at the "degree of satisfaction of a user of the service"[11]. The dominating research perspective on QoS has however, during the years, become more and more a technical one, focusing on monitoring and improving network performance parameters [10] such as the ones listed. A question to be raised is whether these technical network parameters sufficiently reflect the needs of end-users. Taking the example of a web-page download over a lossy network: The end-users do not really perceive the packet loss ratio, but are typically more concerned about the consequences of the packet loss such as a longer page download time. A good point from [8] is that the well-established QoS-tradition is good at investigating *what* is happening in the network but fails in providing insight in the *why*-dimension: *why* does a user behave in a certain way or *why* does the user feel frustrated? This is where QoE comes in, providing several dimensions for understanding the perceived quality. Figure 3.1 illustrates how we place QoS in the bigger QoE picture.

User experience and the concept of QoE was originally promoted by Human-Computer Interaction (HCI) researchers to stress concern with the *outcomes* of people's experience with - or through technology [3].

HCI-researchers tend to emphasize the multi-dimensional character of experience, highlighting the importance of factors such as emotions, expectations, context of use and the relationship to other people [8]. This is in quite a contrast to the "hard" network QoS parameters. Although HCI-researchers have their focus on the multi-dimensional aspect of QoE a great deal of the existing definitions of QoE are rather technology-centric ignoring the subjective character of human experience [8], for example has QoE been defined as: "an extension of the traditional quality of service (QoS) in the sense that QoE provides information regarding the delivered services from an end-user point of view". The ITU-T defines QoE as: "the overall acceptability of an application or service, as perceived subjectively by the end-user"[13]. In any of the definitions QoE aims to grasp a wider area than the network performance-centric QoS. Some researchers has taken this definition even further and wider and state: "Quality of Experience deals with 'all relevant aspects that define how satisfied a person is with a service"'[8]. This definition comes from Kilkki which also, on LinkedIn, states *"Quality of Experience includes everything that really matters"*. QoE focuses on understanding overall human quality requirements which is a multidisciplinary field based on cognitive science, social psychology, economics and engineering science [18]. In the proceeding sections we will relate these QoE and QoS dimensions more directly to video streaming.

3.3 Video Quality

There are many factors influencing and contributing to video QoE. All the processing steps, see Figure 3.2 on the next page, from the actual filming, video post processing, encoding/compression, transmission, decoding to the rendering of the video have strong and direct impact on what actually is seen. Quality degradation of the video may happen in all steps in this process. In addition to this, as described in the previous section, the user is also affected by other factors such as emotions, previous experiences, context of use etc. In [35] a list is provided to enlighten some of the numerous factors affecting QoE:

- Display type (CRT, LCD, iPad etc.) and properties (size, resolution, brightness, contrast, response time, color).
- Viewing setup and conditions (distance, lighting).

Figure 3.2: Video processing steps.

- Video experience of the viewer, will also determine quality expectations (it is hard to go back to standard definition once you have seen high-definition content).
- Individual interests of the viewer (favorite TV-shows etc.) determine the focus level of attention.
- Interaction with the service and the display device (web-interface, remote control, zap-time, EPG)
- Quality and synchronization of the accompanying audio.

The subjectivity and large variety of these factors implies that measuring and optimizing the quality of video systems is a highly complex problem [35]. One can roughly divide between technical and non-technical quality influence factors. The technical ones affect the visual fidelity of the video in terms of distortions in the temporal and spatial video structure introduced by various the processing steps (mainly compression and transmission) [35]. Even if one is just focusing on the technical factors the video systems are still complex (consisting of switches, routers, capture and display hardware, converters and codecs, and all of them process the video somehow and hence may affect its quality); in addition to this, one has to understand how people perceive video and its quality. The visual perception is even more complex [35]. Studies have for instance shown that colorful, sharp pictures with high contrast are considered as more attractive to people than dark, blurry pictures with low contrast [34]. Further discussion on how quality assessment is done is given in the next section.

3.3.1 Video Quality Assessment

When it comes to quality assessment methods a distinction is usually made between *subjective* and *objective* methods [8].

3.3.1.1 Subjective Quality Assessment

The International Telecommunication Union (ITU) defines QoE as "The overall acceptability of an application or service, as perceived *subjectively* by the end user". This definition implies for measurements of QoE to be done through subjective experiments. According to [35] subjective experiments represent the most accurate method for obtaining quality ratings. In subjective experiments a number of people are usually asked to watch a set of video clips for later on to rate what they have seen and experienced. The rating is often done in accordance with the Mean Opinion Score (MOS), which originates from the voice domain as a subjective measure of voice quality [12]. Test users are asked to rate quality parameters using a standardized five point scale with labels such as *Excellent, Good, Fair, Poor and Bad*. Since this is in the subjective domain, one has to expect some variability of the users' ratings as people have different interests and expectations for the video. One can minimize such factors through precise instructions, training and controlled environments [35]. The ITU suggests standard viewing conditions, assessment procedures, criteria for selection of test users and material and data analysis methods.

Subjective quality assessment methods

There exist a variety of subjective quality assessment methods. Recommended methods [35] include Double Stimulus Impairment Scale (DSIS), Double Stimulus Continuous Quality Scale (DSCQS), Single Stimulus Continuous Quality Evaluation (SSCQE) and Absolute Category Rating (ACR). Common for these methods is that they are conducted in a closed lab-environment. The choice of method depends on what you want to measure and the application characteristics (quality range etc.). Details for these methods are omitted from this report as it is out of the scope of the thesis.

For quality assessment in a more natural setting there are several methods available in the field of social sciences. Amongst them are methods such as in-depth interview, survey, observation, etc [8]. The diary method, which is used for the "self-recording of everyday life events" has an advantage [8] as it can capture the particulars of an experience not possible using more traditional methods.

Limitations with subjective quality assessment

As already mentioned, the MOS scale is often used for rating the quality. The use of MOS for quality assessment has however been criticized by authors from various fields [8]. It has for instance been argued that the intervals of the MOS scale are problematic. They are unequal at the conceptual level in addition to that users usually lack the incentive to report accurate scores [23, 8]. Also, due to cultural differences in interpretation, results from subjective experiments are not internationally representative [8].

Most subjective experiments take place in a controlled lab environment, which means that results from different experiments can more easily be compared. On the other hand, as test users are outside their "natural habitat" in such experiments, the external validity of results obtained is highly questionable [8].

The main shortcoming with subjective experiments is, according to [35], the requirement for a large number of viewers, limiting the amount of video material that can be evaluated in a reasonable amount of time. A question raised in [3] is whether one can rely on test users' self-reported opinion when many of the user behaviors enabled of networked services are associated with unconscious psychological factors. This is an argument for more objective quality assessment, which is discussed in the next section.

3.3.1.2 Objective Quality Assessment

Objective quality measurements are not dependent on test users' opinions, but are rather based on direct measures of some process or outcome of user behavior. Objective measurements are typically technology-centric where data is automatically collected by monitoring tools [3]. User tests are then necessary to identify and validate the relationship between technical parameters and the perceived quality (QoE). This section will start out broad and look at what is done in this area with regards to objective quality metrics for video over IP before going into more details on video over TCP and finally DASH, see Figure 3.3 on the following page. We will end this section by looking at user-centric objective quality metrics.

Video over IP

The engineering approach for video quality assessment is primarily based on the extraction and analysis of certain features and artifacts in

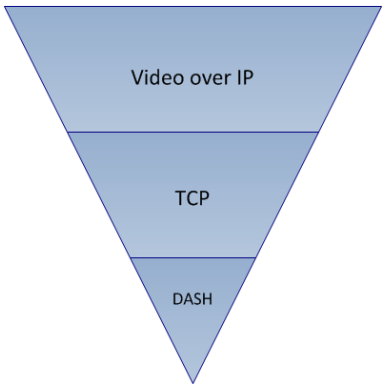

Figure 3.3: Top-down approach for objective quality assessment.

the video. In the image and video processing community, this typically means looking for blockiness, blurriness and jerkiness etc. which are artifacts introduced to the video in the compression/encoding or transmission process [35]. Another approach is to measure the difference between the transmitted video and the original video. Peak Signal-to-Noise-Ratio (PSNR) and Mean Squared Error (MSE) are well-known measurement techniques aiming to measure this video fidelity [35]. These formulas have been popular because they are simple to understand, and fast and easy to compute. PSNR is a byte-by-byte comparison of the data, and thereby only represents an *approximate* relationship with the video quality perceived by human observers [35]. Figure 3.4 on the next page demonstrates this with an example showing two images with identical PSNR at the same time as their perceived quality is very different.

In [35] it is argued that the network QoS community has equally simple metrics quantifying transmission errors such as packet loss rate and bit error rate. Such metrics are relevant to data links, but does not necessarily reflect how the video quality is affected. Excellent QoS does not necessarily mean excellent QoE. Relating traffic characteristics to user satisfaction is not a simple task [28]. There is a growing interest in this field as a result of increasing video service delivery over IP networks [35], as also discussed in Section 2.1.

Video over TCP

DASH systems use, as mentioned in Section 2.2.2, TCP for its data transmission. TCP is optimized for accurate and reliable delivery rather than timely delivery. The reliable features of TCP free the video codec

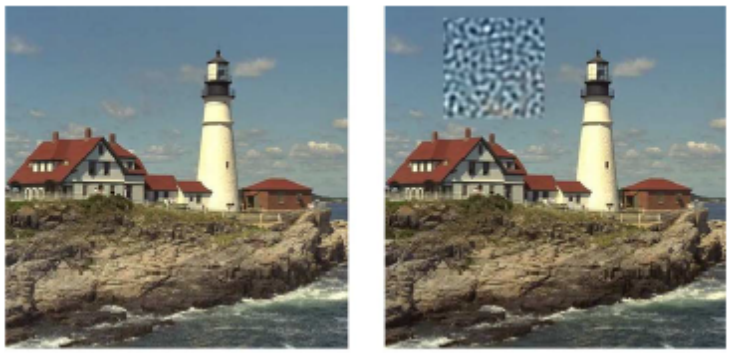

Figure 3.4: The left and right picture have the same PSNR at the same time as the perceived quality is very different. In the leftmost picture the distortions are in the lower area while in the rightmost picture the distortions are in the sky and more easily seen. Figure from [35] (figure 1).

from handling packet loss and the resulting picture quality is not degraded due to missing frames [22]. The frames simply will not be missing. Existing objective measures such as PSNR are hence more suited to quantifying the performance of compression algorithms [27]. These measurements can be very useful when UDP is used for transport of the video stream [27], but the focus of this thesis is on HTTP adaptive video streaming which utilizes TCP.

If the TCP throughput is reduced to a lower level than the current playback rate, the buffer will be utilized and eventually become exhausted. This causes the video playback to stop, and such a disruption could greatly impact the perceived QoE. For this reason the authors of [22] propose to focus on the temporal structure of videos streamed over TCP. The running application (the video player) is directly affected by network QoS, and the users are in turn affected by the video player's performance. They further on propose three application performance metrics:

1. Initial buffering time
2. Mean duration of a rebuffering event.
3. Rebuffering frequency

Through subjective experiments in [22] it is revealed that the rebuffering frequency is the main factor responsible for variations in QoE. Porter and Peng in [27] have more or less the same approach, but combine the rebuffering frequency and duration into one metric which they call *pause intensity*. Results from their subjective test show that the pause intensity and the subjective scores are closely correlated. Another interesting finding in [22] was that the effect from the initial buffering time was not significant and implying that users are generally willing to tolerate a longer startup delay for a better video watching experience. It should be noted that the experiments conducted by Porter and Peng were only simulations of a *progressive* HTTP video stream with one quality level, and also the study by [22] was focusing on *progressive download*. Progressive download was described in Section 2.2.3. Their studies has however shown that the temporal structure, instead of spatial artifacts (the picture), is an important factor affetcting the QoE for HTTP video streams.

DASH

DASH video streaming enhances the QoE for users by its automatic switching between quality levels when, for instance, there are fluctuating network conditions. Quality transitions are particularly relevant to DASH-systems, and wrong decisions on choosing quality level may have a huge impact on the perceived quality. In [24] subjective experiments have been conducted to measure the effect such quality transitions have on QoE. In the tests it was found that inserting intermediate levels between quality drops was favorable for the perceived quality, compared to switching directly to the target quality level. Another interesting finding was that in one of their test scenarios the video with the lowest quality level throughput did *not* obtain the lowest MOS. In this case there were no quality transitions, and the users did not know that they were watching the video with the lowest quality. These experiments indicate that a stable, but lower quality level is better than having a stream with high frequency of quality transitions even though it, on average, provides a higher quality level. In [32] another metric related to quality transistions is introduced, calling it *perceived fairness*. The authors of [32] relate the metric to the social science and psychology domain where it has been found that users react negatively to any system behavior which gives better service to another user, unless justification is provided. In terms of quality of adaptive video streams this can be translated to that end-users may perceive certain system behavior as un-fair. This could for instance

be quality transitions to a lower level, frequent quality transitions or periods with the session rate below some threshold. It has also been found that quality perception is asymmetrical when adapting the quality down and when adapting the quality up [7]. Users are more critical to quality degradations and less pleased by increased quality.

User-centric objective quality assessment

We have now focused on technology-centric objective quality measurements. According to [3] there is a widespread misunderstanding that objective quality measures can *only* be collected from technology and that data from users are necessarily subjective. Some research in this area [23] has investigated the correlation between user-viewing activities, network path performance and QoE. Low TCP throughput can, as previously explained, cause buffer exhaustion and in turn stop the video playback, thus destructing the temporal structure of the video playback [23]. Through user-activities such as pausing and reducing the screen-size the negative impact of a low TCP throughput can be mitigated. Results from the survey in [23] show that users choose pausing, switching to a lower video quality and watching with a normal screen size under a jerky playback scenario. Measuring such user-activities can thus give an indication of network impairments and bad QoE. Indicators of user satisfaction can also be found from network traffic traces. User experience described through the interruption probability of user HTTP connections has been presented [28].

Objective measures of QoE *can* and *should*, according to [3] also be collected from user tests as these measures enables us to extend beyond *user perception* to *user experience*. The point is that measures of QoE should concern user performance and based upon actual usage. In terms of video streaming this could be person-content interactions such as mouse clicks, time between mouse clicks, eye-tracking, task completion time, et cetera.

3.4 Why Measure Quality?

Better network QoS will in many cases result in better QoE, but fulfilling all traffic QoS parameters will not guarantee a satisfied customer [25]. The test scenario (mentioned in Section 3.3.1.2) from [24] where the video sequence with the lowest throughput did not obtain the lowest MOS is an example of this. From a service provider's perspective, having fo-

cus on customer satisfaction is important as the risk of customer churn is high in a competitive market. In a survey done by Accenture [25] it is shown that, on average, one dissatisfied customer will tell 13 other people about their bad experiences. From the amount of customers having experienced troubles only one out of 30 will report their problem to the service provider [25]. A way to avoid losing customers is to constantly measure QoE and improve it as and when needed. Constantly measuring QoE requires good, objective quality parameters, which can relate potential network impairments and other factors to end-user QoE.

Reasons to motivate the interest for QoE, in addition to user satisfaction, include network configuration and design, and service testing [14]. In [26] it is also pointed out that quality can be the differentiator when other factors are equal, if the price is the same but company X have a reputation in always delivering a high-quality service, people are then likely to choose company X. It is also possible to price the service differently according to different quality levels. This could be an instrument for reaching a bigger audience as people may have different expectations and willingness to pay for Internet TV. Comoyo, with their Video on Demand (VoD) service, is an example of an actor doing this. On some of their movies, customers can choose to pay more to have access to High Definition (HD) content rather than Standard Definition (SD).

Ensuring user satisfaction is, as earlier explained, important to the service provider to keep their market position and customers. Quality measurement is important to a large variety of stakeholders. In Figure 3.5 on the facing page an overview is given of the different stakeholders for QoE data.

Measuring subjective and objective user experience metrics in a quantitative way will ensure that a large range of techniques can be applied afterward to derive QoE ratings. This is important so that one can communicate QoE in an optimal way to the various stakeholders [3].

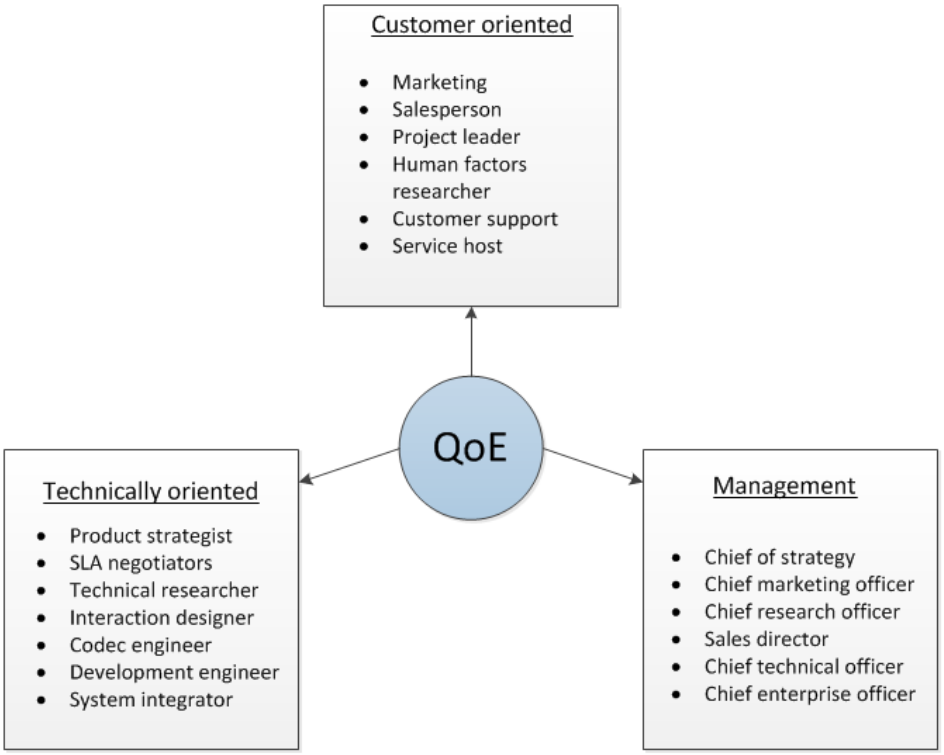

Figure 3.5: QoE data stakeholders with different roles and backgrounds. Figure adapted from [3] (Table 3).

3.5 Quality Metrics for Adaptive Video Streaming

In the preceding sections we have provided a foundation for understanding the QoS and QoE dimensions of adaptive video streaming. We have seen that the dominating research perspective on QoS has during the years become more and more a technical one focusing on improving and monitoring network performance. While QoS parameters can provide good insight in *what* is happening in the network, QoE can also provide insight in the *why* dimension: *Why* does a user feel frustrated? QoE provides several dimensions for understanding user experience. In the following we will discuss the quality metrics applicable for adaptive video streaming which relates both the QoS and QoE dimensions.

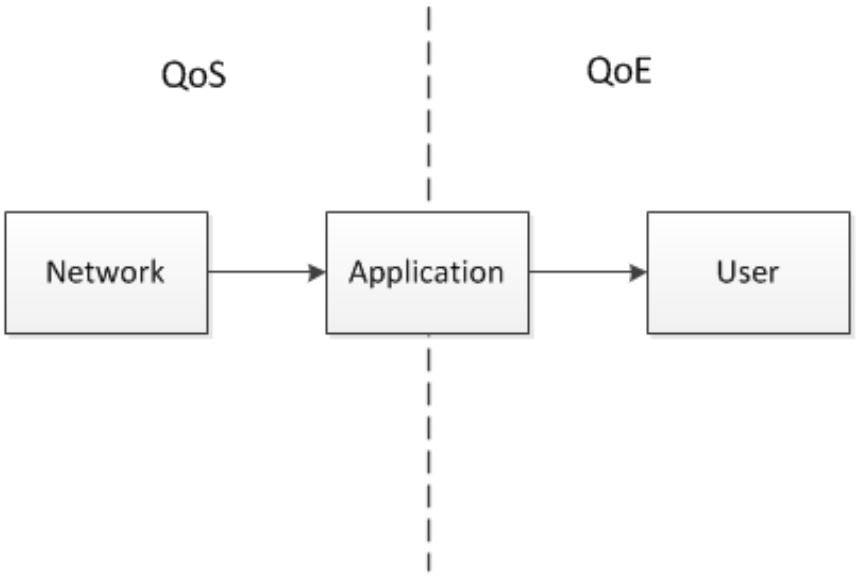

Figure 3.6: Bridging the gap between QoS and QoE.

One way to relate QoS and QoE is to look at the application performance metrics. The application (in our case the video player) is, as depicted in Figure 3.6, directly affected by network QoS and has at the same time direct impact on end users' QoE. In [22], three application performance metrics for HTTP video streams were proposed: (1) *initial buffering time*, (2) *mean duration of a rebuffering event* and (3) *rebuffering frequency* (for more details see Section 3.3.1.2). Network QoS parameters have a direct

impact on these metrics. The three metrics are also applicable to adaptive video streaming. Metric (2) and (3) are although likely to happen less frequently in DASH systems compared to progressive download as a quality rate reduction would normally be triggered before a rebuffering event occurs.

What makes DASH systems special are their automatic switching between quality levels. *Quality transitions* is absolutely a quality metric to consider. A study [24] supporting this statement has shown that end users could prefer a stable video stream (fewer quality transitions) at the expense of an overall higher bitrate. In this study it was also found that inserting intermediate quality levels when degrading the quality provides a better QoE than if switching directly to the target quality level. It has also been found that quality perception is asymmetrical[7]: users are more critical to quality degradations and less rewarding to increased quality.

Another metric is the available *bitrates* (quality levels) at the hosting webserver. The higher bitrate the better. However, this metric should be considered in relation to the quality transition metric, as just described.

3.5.1 Scope

The metrics described are all objective application performance metrics which from various subjective studies have shown to have impact on end users' QoE. The finding that a higher bitrate does not always lead to a higher degree of QoE (when it comes at the cost of more quality transitions) might not be very intuitive. To see if this is considered by today's adaptive video streaming technologies we will perform two case studies by means of measurements on the video players of Tv2 Sumo and Comoyo.

The focus will mainly be on the players' reaction to packet loss, whereby we will introduce a sufficient amount of packet loss forcing the players to react. Among the questions we want answeres to are: How long time will it take before the player triggers its first quality transition? Will the player stabilize at some quality level, or will we see many quality transitions throughout the session?

Tv2 Sumo's and Comoyo's player are both based upon Microsoft's Smooth Streaming technology. However, as explained in Section 2.2.4.2, this does not mean that they are identical since developers are free to

modify the Silverlight player. We therefore find it interesting performing measurements on them both.

Chapter 4

Measurements

This chapter starts by clarifying the main objectives for performing measurements on Comoyo's- and Tv2 Sumo's video player in Section 4.1. A lab has been set up for the collection of data and is described in Section 4.2. In Section 4.3 we define the measurement scenarios, and in Section 4.4 we describe how the measurements are to be performed. In the last section of this chapter we describe how we process and present the data.

4.1 Objectives

In Section 3.5.1 we gave motivation for performing measurements on Comoyo's- and Tv2 Sumo's Silverlight player. The main objectives are to investigate how packet loss affects the video player and to see if the video player takes *quality transitions* into consideration together with *bitrate*. Through the measurements we will try to answer the following questions:

1. Will the player stabilize at some quality level, or will we see many quality transitions throughout the session after the introduction of packet loss?
2. How much time will pass before the player triggers its first quality transition?

Answers to these objectives can give us an indication to whether quality transitions is a metric considered by the players.

Motivation for objective one:
If the player after some time stabilizes under varying network conditions, not always striving for the highest quality level, we consider it as a sign that the player in fact does consider quality transitions as a quality metric in relation to bitrate.

Motivation for objective two:
If the player is exposed to short-term network fluctuations we would like the player to rather utilize a bit more from its playback buffer so that unnecessary quality transitions can be avoided. When we study the players' reaction time to packet loss we consider it positive if the player use a long time before it first degrades the quality.

4.2 Data Collection

4.2.1 Lab set-up

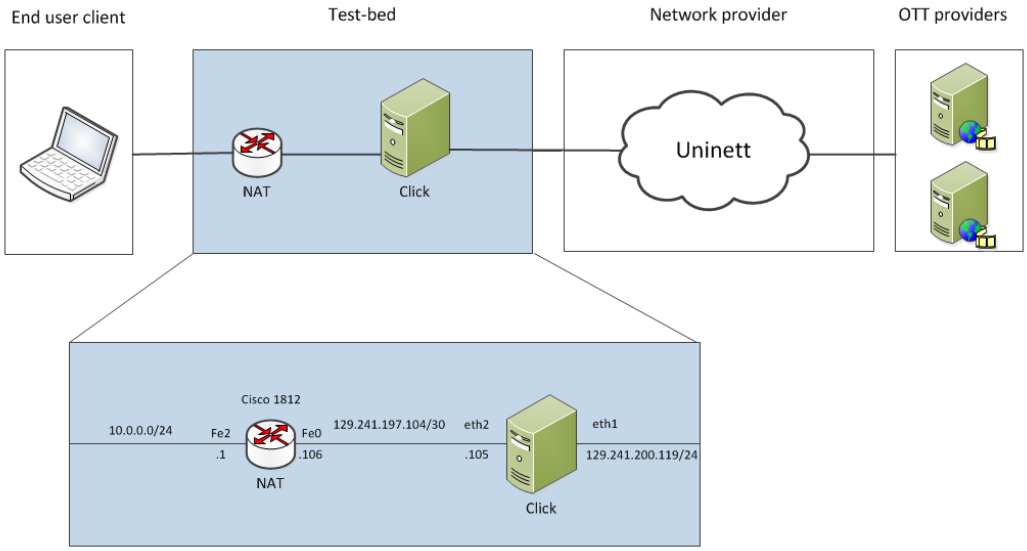

Figure 4.1: The lab environment.

To collect the wanted video traffic data a lab has been set up as shown in Figure 4.1. The lab consists of an end user client, a Cisco router and a Click modular router, which are decribed in the forthcoming subsections.

4.2.1.1 End User Client Specification

The computer used as the end user client has the following specifications:

- Windows 7 Professional SP1 64-bit
- Intel Core i7-860 @2.80GHz 2.93 GHz
- 4GB RAM

4.2.1.2 Cisco Router/NAT

The router is a Cisco 1812 Version 12.4.

On the Cisco router we can apply rate limitation through QoS policy maps. Two policy maps have been defined on the router, one for 10 Mbps and another for 20 Mbps bandwidth limitation:

```
policy-map 10MB
    class CLASS_SLAP
        police 10000000 1000000 conform-action transmit
        exceed-action drop


policy-map 20MB
    class CLASS_SLAP
        police 20000000 1000000 conform-action transmit
        exceed-action drop
```

The rate limitation policy is activated with the following commands when logged in on the router:

```
conf t
interface FastEthernet0
service-policy input 10MB
exit
exit
wr
```

4.2.1.3 Click Modular Router

The specifications of the Click Modular Router:

- Ubuntu 8.04 32-bit
- Intel Premium 4 2.80GHz
- 3GB RAM

The Click router runs on a Linux machine which runs a clickpatched version of Ubuntu 8.04. This means that there are some modifications made on the OS kernel to better suit Click. Click is an open source project developed at MIT and is a software architecture for building flexible and configurable routers. A Click router is assembled from several packet processing modules called *elements* [15]. The various elements implement simple router functions like queuing, packet classification, scheduling and interfacing with network devices. We utilize Click to introduce packet loss while streaming video. What we basically do is to inspect each incoming TCP packet and drop it with a certain probability. In Click this means that we use an *element* to classify incoming TCP packets, this can be done using the IPClassifier element [15]:

> **IPClassifier(...)**: The input takes IP packets and packet data is examined according to a set of classifiers, one classifier per output port. Forwards packet to output port corresponding to the first classifier that matched. Example classifier: "ip src 1.0.0.1 and dst tcp port www" checks that the packet's source IP address is 1.0.0.1, its IP protocol is 6 (TCP), and its destination port is 80.

As we are interested in the incoming TCP packets our utilization of the IPClassifier element looks like this:

```
find_data_eth2 :: IPClassifier(tcp, -);
```

The TCP packets are later on dropped with a certain probability:

```
-> RandomSample(DROP 0.05)
```

For more details on the Click configuration see Appendix B.

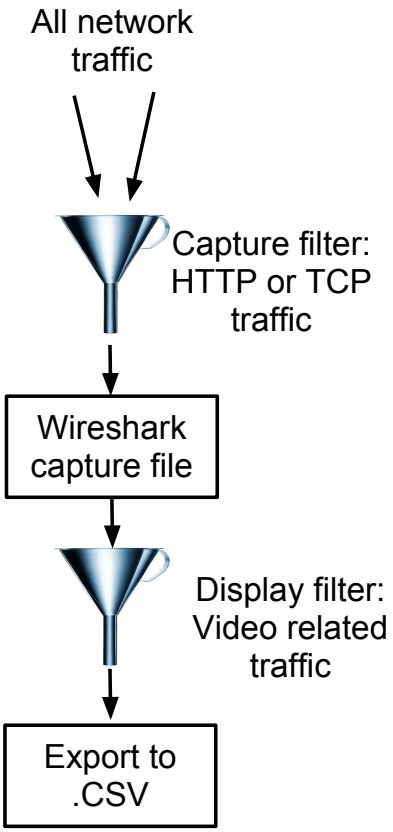

Figure 4.2: From raw network data to a video trace file.

4.2.2 Traffic Measurement

There are many applications which can be used to monitor and record traffic coming in and going out of the computer's network interface. Wireshark (described in Appendix A) is one of the most used applications for this purpose, and is also the tool we have utilized for our measurements. Wireshark is run at the end user client machine. Capturing all incoming and outgoing packets from the network interface is storage and resource expensive, to reduce this load capture filters can be applied. In addition to capture filtering one can apply display filters on the capture file for post processing. Figure 4.2 shows us how this filtering is done in our measurements.

Recall from Section 2.2.4 that video segments are fetched from the hosting webserver using the HTTP GET method and is transported by

TCP. This means that for capturing the video traffic we only need to look at TCP and HTTP traffic. This is done by setting the capture filter in Wireshark to:

```
tcp port http
```

The capture file (a .pcap file) we end up with should contain all HTTP and TCP traffic passing through the network interface in the monitoring period. Further on we use Wireshark's display filter to only assess the video related traffic. The assessment of this traffic is done by first finding the hosting webserver's IP address for later filtering out the traffic going to and coming from this address. This can be done by looking at the destination address for the GET-requests sent to the webserver. It has, however, been noticed that for some of the streamed sessions this destination address occasionally changes, probably due to some load balancing at the server side. We therefore first check whether or not this is a case before filtering out traffic for one or several IP addresses. We have performed this check by filtering out video GET requests and looking for multiple destination addresses. The filter applied:

```
http.request and expert.message contains "video"
```

When the hosting webserver's IP address (or addresses) has been found we filter out the traffic coming from and going to this address:

```
ip.addr == 148.122.38.92 (example address)
```

We now have a complete trace of the streamed video session and for further processing we export the file to a Comma Separated Values (CSV) file. The generated .csv file is based upon the displayed columns in Wireshark, in our case:

- **No.** - the sequence number of the packet in the capture file.
- **Time** - the timestamp of the packet, in our case number of seconds since beginning of the capture (with nanosecond precision).
- **Source** - the address from where the packet is coming.

- **Destination** - the address to where the packet is going.
- **Protocol** - the protocol name (HTTP or TCP) .
- **Length** - packet size (in bytes).
- **Info** - additional information about packet content.
- **Src port** - port number for where the packet is coming from.
- **Dest port** - port number for the packet destination.

These elements are exported to .csv. The procedure from the collection of raw data to the generation of the .csv file is repeated for all experiments.

4.2.3 Limitations

The lab set-up described in the preceding sections has worked quite well. In this section, however, we will describe the limitations and problems we have experienced so that further improvements can be made.

4.2.3.1 Monitoring at the End Node

As explained in Section 4.2.2, the recording and monitoring of the video traffic is done by Wireshark at the end user node. As Wireshark can be quite resource expensive, a possible scenario is therfore that Wireshark might interfere with the measurements, thus causing the generation of faulty data. The best solution would be to have an intermediate node by which all video traffic is sent for post processing. Due to the time limitations of this thesis this has not been done. However, we believe that this scenario is not applicable for our measurements as we only perform measurements on one video stream at the time.

4.2.3.2 Random Packet Loss

In Section 4.2.1.3, we described that the Click router drops incoming TCP packets with a certain probability. A question to be raised is how realistic this scenario is, and as future work of the thesis one could consider emulating a more realistic packet loss scenario (bursty packet loss, etc.).

4.2.3.3 Wireshark Problems

We experienced several problems with Wireshark when starting the measurements. The problems started while monitoring traffic from tv2sumo.no; there were periods with no video traffic recorded, although the video was playing continuously in the background (silent periods up to 60s were observed). Alternatives to Wireshark were therefore investigated: among them Capsa, Fiddler and dumpcap. These alternatives did not provide any better results as we still saw parts of the video traffic missing. We also tried performing measurements on other end clients than the one specified in Section 4.2.1.1, but with no luck.

However, when performing measurements on the Tv2 Regular live sendings on a video player stripped for everything but the player (this player can be found on cap.item.ntnu.no) we did not see the same large fractions of missing video traffic. So we decided to perform the measurements on the live sendings at the cost of not being able to perform repeated measurements on the same video sequence. The same problems were experienced while doing measurements on Comoyo's player: fractions of video traffic were missing in the capture file. A semi-solution to this was found by doing jumps in the video to a random place, this seemed to "wake" Wireshark up again.

Due to the explained problems the measurements have taken much more time than necessary.

4.3 Planning Scenarios

In Section 4.1 we defined the objectives for performing measurements on TV2 Sumo's and Comoyo's Silverlight players. However, before starting the measurements we need to decide upon exactly *how* this is to be done. The questions to be answered include:

1. What video content are we going to perform the measurements on?
2. Which packet loss ratio should we use, and should there be any bandwidth limitation?
3. For how long time should we measure and how many measurement runs are needed?
4. How much data will have to be processed?

These questions are addressed in the next subsections.

4.3.1 What Will be Measured?

Although the 2s video segments are encoded at the same QL they vary in size. In the next chapter a graphical visualization of this is shown. Performing repeated measurements on the same video sequence could therefore be beneficial, as we would not need to take these variations into account in the analysis phase. However, this has proven to be difficult to do in our case for two reasons:

1. We are considering two different service providers (Tv2 Sumo and Comoyo).

2. Problems with Wireshark.

The two different service providers offer different video content; therefore, performing measurements on the same video sequence is not possible. We have also repeatedly experienced problems with Wireshark, as described in Section 4.2.3.3. This has caused the measurements performed for Tv2 Sumo to be performed on their live sendings "Tv2 Regular" from the player on cap.item.ntnu.no. Live sendings means different content for each measurement. The problems with Wireshark were also experienced when performing measurements on Comoyo's player. A way to cope with this problem was found by jumping to a random time in the video, this seemed to wake the Wireshark process to measure the incoming traffic. Jumping in the video to a random time, however, makes it impossible to perform repeated measurements on the same video sequence.

After these considerations, we to perform measurements on Tv2 Sumo's live sending, which offers the following quality levels: {0.25, 0.75, 1.5, 2.5, 3.5, 5.0} Mbps (found in the manifest file in the Wireshark capture). "Varg Veum - I mørket er alle ulver grå" is the movie chosen for the measurements performed on Comoyo's player. It was chosen because its highest QL were the one closest to the highest QL of Tv2's live sending. The offered bitrates of the Comoyo movie are {0.097, 0.197, 0.295, 0.493, 0.790, 1.177, 1.578, 2.178, 3.959, 5.934} Mbps.

PL ratio	Sumo	Comoyo
2%	No QL reduction	No QL reduction
4%	No QL reduction	QL reduction
5%	QL reduction	QL reduction
7%	QL reduction	QL reduction

Table 4.1: Test runs.

4.3.2 Packet Loss Ratio and Bandwidth Limitation

As we are interested in observing the video player's reaction to packet loss, we need to find under which circumstances the player will trigger a QL degradation due to packet loss. For this reason, we have, performed some test runs with different configurations of the Click Router (which is, as explained in Section 4.2.1, the entity dropping incoming packets with a certain probability). The packet loss in the test runs was first introduced after the player has already been running for a while, so the player is already running at the highest QL and we assume the player to be in a steady state (more on this in Section 5.2). Table 4.1 show the results for the different test runs.

From Table 4.1 we can see that when 2% packet loss is introduced neither Sumo nor the Comoyo player react. With 4% packet loss Sumo is able to keep the highest QL, while Comoyo's player reduce the requested QL. On 5% and 7% packet loss QL reduction is triggered in both players. It should be noted that these testruns were run only to get an indication for which packet loss ratio a reaction from the two players was triggered. It was thereby decided to continue the measurements with a packet loss ratio of 5% and 7% in addition to performing measurements with no packet loss to get an impression of how it "should be".

As explained in Section 4.2.1.2, a bandwidth limit can be set on the Cisco router through the defined policy maps. We have chosen to perform the measurements on the 10 Mbps profile to get a bit more realistic testing scenario compared to no limit. A 10 Mbps bandwidth limitation will not cause any network congestion since the highest QL offered in the video content to be measured is 5.934 Mbps, which is good as our focus is on the effects of packet loss, not congestion.

4.3.3 Measurement Length, Number of Runs and Dataset Size

We will perform measurement runs of 10 minutes, as this allows us to perform the measurements in a reasonable amount of time and long enough to say something about how the two players react to packet loss over time.

To perform the measurements in a reasonable amount of time it was decided to restrict the repeated number of measurement runs to 5. As we are going to perform measurements for three scenarios (no packet loss, 5% loss and 7% loss) for two different players this will result in 2 (players) x 3 (scenarios) x 5 (runs) x 10 (minutes) = 300 minutes of video measurements. To give an indication for the amount of data this generates, the size of the Wireshark capture file from one 10 minute measurement period is roughly 230 000 KB. One .csv file, containing only the video trace, consists of about 2-300000 lines of data. From this data we expect be able to answer the questions raised in Section 4.1.

4.4 Performing Measurements

The measurements for the two case studies (Tv2 Sumo and Comoyo) which includes 3 scenarios (no packet loss, 5- and 7% packet loss) and 5 repetitions are all performed the same way:

1. **Time = X**: Video playback starts.
2. **Time = 0s**: After 1-2 minutes of video playback the Wireshark capture is started, to be sure that the playback buffer is filled up at that point.
3. **Time = 30s**: Click starts to drop incoming TCP packets.
4. **Time = 600s**: End of Wireshark capture.

Step 3 is ignored if it is a no packet loss scenario. It should be noted that the Click router is started manually, so one can expect small variations for the start-up time. Click is started through a shell script called mystart.sh, Figure 4.3 on the following page shows the Click working environment.

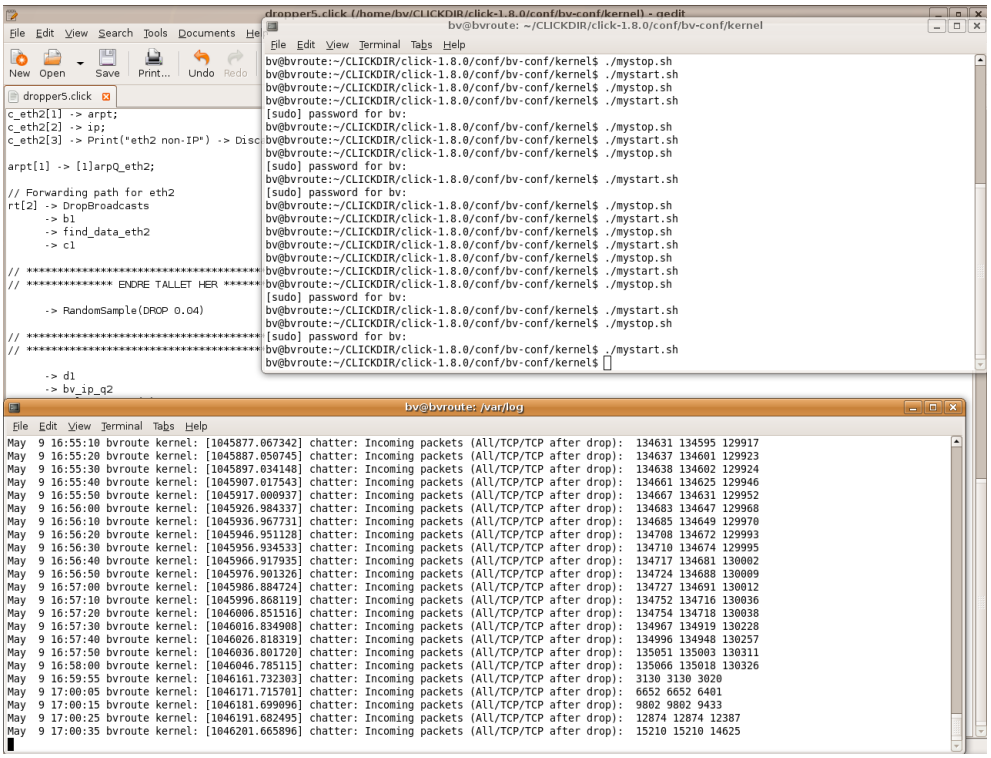

Figure 4.3: A screenshot showing the Click router in operation during one of the test-runs.

4.5 Processing and Presenting Data

After having performed the above described measurements we end up with 2 cases x 3 scenarios x 5 repetitions = 30 *.csv files containing roughly 2-300000 lines of packet information, each. We extract the data of interest in a Java program (the program can be found attached to the report) which takes a video trace file as input. Figure 4.4 shows a trace file sample from one of the measurement runs of Tv2 Sumo.

```
1 "No.","Time","Source","Destination","Protocol","Length","Info","Src port","Dest port"
2 "1147","1.867959000","10.0.0.2","158.38.14.142","HTTP","784","GET /live/324298.isml/QualityLevels(5000000)/Fragments(video=34353500000000) HTTP/1.1 ","54901","http"
3 "1148","1.909897000","158.38.14.142","10.0.0.2","TCP","60","http > 54901 [ACK] Seq=1225880 Ack=1461 Win=12037 Len=0","http","54901"
4 "1149","1.935585000","10.0.0.2","158.38.14.142","HTTP","784","GET /live/324298.isml/QualityLevels(96000)/Fragments(track_1=34353502666666) HTTP/1.1 ","54902","http"
5 "1150","1.940365000","158.38.14.142","10.0.0.2","TCP","1514","[TCP segment of a reassembled PDU]","http","54901"
6 "1151","1.940367000","158.38.14.142","10.0.0.2","TCP","1482","[TCP segment of a reassembled PDU]","http","54901"
7 "1152","1.940398000","10.0.0.2","158.38.14.142","TCP","54","54901 > http [ACK] Seq=1461 Ack=1228768 Win=65335 Len=0","54901","http"
8 "1153","1.940817000","158.38.14.142","10.0.0.2","TCP","1514","[TCP segment of a reassembled PDU]","http","54901"
9 "1154","1.940818000","158.38.14.142","10.0.0.2","TCP","1514","[TCP segment of a reassembled PDU]","http","54901"
```

Figure 4.4: Sample from a video trace file.

When the data of interest has been extracted we write the results to .txt files. Mathematica has then been used for further processing and presentation of the results. The overall procedure from reading in the video trace file to data presentation is shown in Figure 4.5.

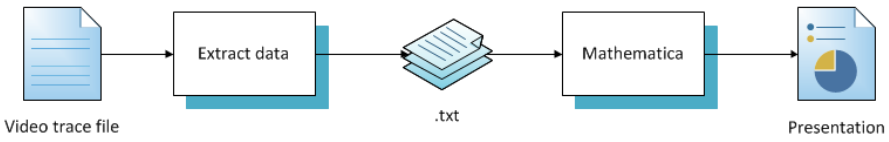

Figure 4.5: Processing workflow.

4.5.1 Extracting Data

For the Java program to extract the data of interest we need to know how the information within the trace file is structured. An overview of the content in the trace files is given in Section 4.5.1.1 and in Section 4.5.1.2 we briefly describe how the data processing and the testing of the program has been performed.

4.5.1.1 Data Structure

What we see in the trace file is basically a reflection of the Microsoft's Smooth Streaming client-server communication protocol, described in Section 2.2.4.2. What happens is that the client sends a HTTP GET request to the server; which responds by sending the requested segment. The server sends a HTTP 200 OK message to the client when the last packet within the segment has been sent.

In the trace files we can see the GET requests, the 200 OK messages and the TCP packets in between. We also observe from the trace files that the Silverlight clients maintain two TCP connections with the server: one for audio and one for video. The two connections have different source port numbers, and it has been observed that the player switches between the two connections at various points in time. This behavior was also noticed in another study [1] where the authors had a theory that this behavior could keep the server from falling back to slow start as neither of the connections would stay idle for a long time.

The content is transferred using TCP, and packets carrying data can be identified in the trace file as they are marked in the info-field as "[TCP segment of a reassembled PDU]". It is possible to distinguish between audio and video content by combining port number information from the video/audio GET request and from the transmitted packets. Retransmitted packets can be identified in the trace from the info-field by looking for "[TCP Retransmission]" and "[TCP Fast Retransmission]".

In addition to this we have noticed that the GET requests are pipelined: a new video segment is only requested after the previous one has been fully received.

Differences between Comoyo and Tv2 Sumo

The address which the clients use to retrieve video and audio segments from is slightly different in Comoyo and Tv2 Sumo. For handling both players we have made some modifications of the processing program.

4.5.1.2 Processing and Testing

Processing

For each trace file (for each measurement scenario) our processing program outputs about 10 files containing different information on the

video traffic characteristics. The data is combined in different ways, in this section we will not describe all details of the data extraction but rather focus on what we look at as the "basic building blocks".

Basic Building Blocks

As will become clear in the in next chapter we have mainly focused on the extraction of the following data:

- Requested quality level.
- Time for video GET request.
- Number of TCP packets within a video segment.
- Segment transmission time (time from the GET request to the 200 OK).

From the above metrics many combinations can be made. However, the requested quality level is probably the most important metric to extract as we in this thesis are focusing on quality transitions in relation to the achieved bitrate (quality level), which are both derived from the requested quality level. Listing 4.1 show how we process the trace file to extract the requested quality level and the corresponding time for the video GET request.

```
for(int i =0; i< inputFile.size(); i++){
  String [] fields = inputFile.get(i).split(",");
  if(fields[6].contains("video=")){
    timeGET.add(fields[1].substring(1, fields[1].
       length()-2));
    requestedQualityLevel.add(findQualityLevel(fields
       [6]));
  }
}
```

Listing 4.1: Extracting the requested quality level and the corresponding time for the GET request.

The inputFile is the video trace file generated by Wireshark. This is read line by line. If fields[6], which is the info-field in the trace file, contains the string "=video" we know that this entry is a video GET request. So we save the time for the GET request in one list and requested quality

level in another list. Recall from Section 2.2.4.2 that GET requests are on this format:

GET /live/324298.isml/QualityLevels(250000)/Fragments(video=16151840000000) HTTP/1.1

The method findQualityLevel(fields[6]) extracts the quality level from a string similar to the one above.

Finding the number of packets within a video segment and finding the segment transmission time is a bit more complicated. In order to obtain these values we first find the time of two successive video GET requests, so as to get a subset (the data between the time of the two GET requests) of the input video trace file. Inside this subset we extract the 200 OK message belonging to the video GET request (by checking the port number, we avoid the audio 200 OK). By finding the 200 OK we now know the segment transmission time, and to assess the number of packets carrying video data within a segment we just count the packets within this time flagged with [TCP segment of a reassembled PDU] for the same port number as the video GET request. For more details we refer the reader to the attached code.

Output

As mentioned many files are created by this program. Many of them are formatted such that not much more processing is required. This typically means long lists of data which are easily read and processed by Mathematica.

Testing

We have not developed a own testing framework for this program; however, after every source code modification (the extraction of some new data) we have been taking several samples of the result and checking whether it corresponds with what we can see from the original trace file.

4.5.2 Mathematica

We have utilized Mathematica for drawing graphs for many of the calculations shown in the result. Mathematica takes the files created by the Java program as input. In Appendix D one of the modules made for plotting graphs is shown in order to give the reader an impression on how we have utilized Mathematica.

Chapter 5

Results

For both Tv2 Sumo and Comoyo we have performed measurements on three scenarios:

1. No packet loss
2. 5% packet loss
3. 7% packet loss

The measurements are performed in order to answer the objectives and goals described in Section 4.1. In this chapter the results from the measurements are presented.

5.1 Methods, Definitions and Assumptions

Packet payload:

Each data packet carries about 1500 bytes of information; this includes approximately 40 bytes of overhead. When performing calculations including packet size we therefore estimate the payload of each packet to be 1460 bytes.

Active and silent period:

Video segments are typically requested once every 2 seconds when the player is in its steady state. The time between two successive video GET requests can be divided into an “active” and a “silent” period where the

active period is during the segment transmission and the silent period is when no video is being transmitted, see Figure 5.1.

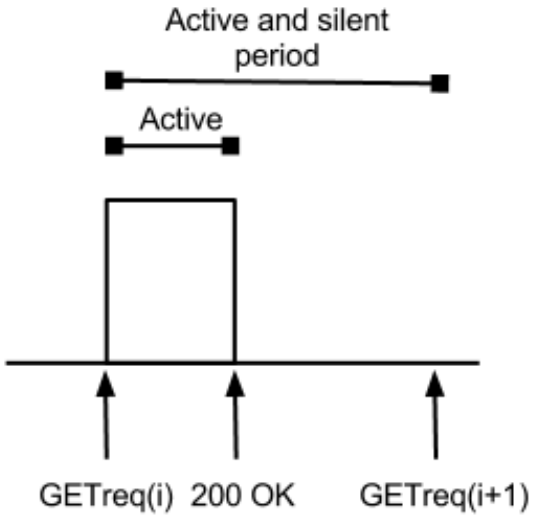

Figure 5.1: The time between two successive video GET requests can be divided into an active and a silent period.

Segment goodput:

The segment goodput is derived by summing the payload of all TCP packets within a video segment divided by the transfer time (the active period).

Average goodput:

The average goodput refers to the moving average of the 2s segment goodput measurements. This metric takes into account the whole interval between two video requests in contrast to the *segment goodput* which only takes the active period into account. Assume that a video GET request is sent at time t_i and that the total number of received TCP packets containing video content at time t_i is given by the variable $numTCP(t_i)$, then the moving average ($A(t_i)$) of the segment goodput at time t_i is:

$$A(t_i) = \frac{payload}{t_i - t_0} * \sum_{i=0}^{i} numTCP(t_i) \tag{5.1}$$

Buffer usage estimation:
The buffer usage is estimated using the video GET requests' timestamps in addition to the fact that between two video GET requests 2 seconds of video is received. Suppose that two successive video GET requests are sent at times t_{i-1} and t_i. Their time difference is what goes out of the playback buffer. The playback buffer's state (in seconds) is estimated as shown in equation (5.2) and (5.3).

$$B(t_i) = B(t_{i-1}) + input - output \tag{5.2}$$

$$B(t_i) = B(t_{i-1}) + 2s - (t_i - t_{i-1}) \tag{5.3}$$

This method will work because video requests are pipelined: a new video segment is only requested when the previous one is fully received. However, as we do not know $B(t_0)$, which is in our case the buffer size when the buffer is full, we can only be certain about the buffer usage. The buffer usage would be the difference between $B(t_0)$ and $B(t_i)$. In the following sections we have used $B(t_0) = 10s$, just to have some reference value.

5.2 No Packet Loss

In this section we will provide an overview of some of the characteristics of the two players (Tv2 Sumo and Comoyo) when no packet loss is introduced in the network. This is of importance for the understanding of how the players work when they are not affected by the randomized packet dropping entity. We start by looking at Tv2 Sumo in Section 5.2.1, thereby we continue with Comoyo in Section 5.2.2.

5.2.1 Tv2 Sumo

Available quality levels: {0.25, 0.75, 1.5, 2.5, 3.5, 5.0} Mbps.
Video: Tv2 Regular, live.

To get an overview of what is going on in both the video player and in the network we have plotted the requested QL, segment goodput and average goodput in Figure 5.2 on the following page. The figure shows that the requested QL is stable at the highest level (5 Mbps). This was expected as our bandwidth limitation is 10 Mbps, so we should be able to stream video at a QL of 5 Mbps. The segment goodput varies quite a

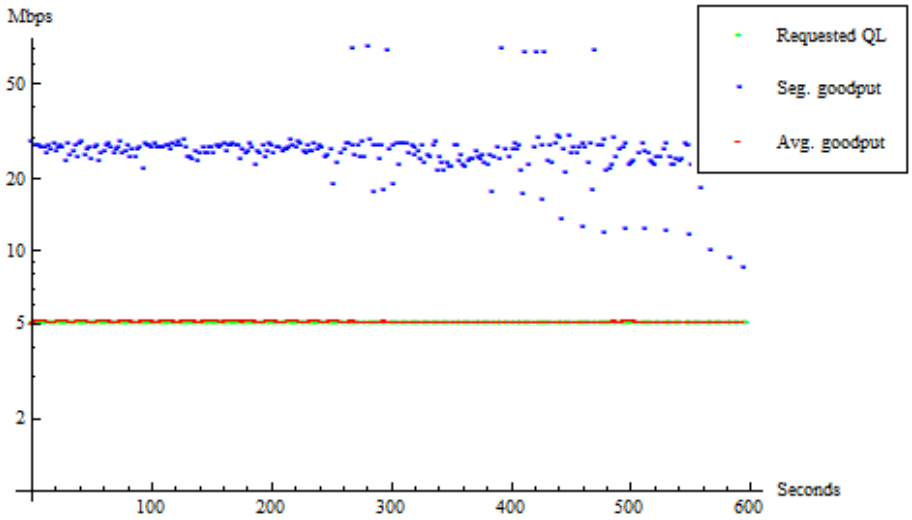

Figure 5.2: A plot showing the segment goodput, requested quality level and average goodput for one run of Tv2 Sumo's no packet loss scenario. It should be noted that the y-axis is logarithmic.

bit, but most segments lies around 25 Mbps. This implies that the segments are transferred quite fast with a short active period and a longer silent period. The red line is the moving average of the segment goodput, defined in Section 5.1. As the requested video segments are all from the 5 Mbps QL we would expect that the average goodput to be more or less stable around 5 Mbps, which we can observe in Figure 5.2. A reason for the slight variations in the average goodput (the red line in Figure 5.2) could be variations in the number of packets in a 2s video segment. With a 2s video segment encoded at 5 Mbps, assuming a packet payload of 1460 bytes, one can expect the number of packets in the segment to be around $\frac{5000000bps*2s}{1460*8bit} \approx 856$. Figure 5.3 on the next page shows that this is the case for our measurements, although we have observed segment sizes spanning from 700 to over 1000 packets within a 2s segment.

The Wireshark capture was started after having streamed the video for 1-2 minutes, so we assume the player to be in a steady state. With steady state we mean that video segments are requested at about the same rate as the video playback rate, which implies that the playback buffer is stable and filled up. In Figure 5.4 on the facing page the time between GET-requests is plotted, showing that, on average, a new 2s video segment is requested every ≈ 2 seconds. However, from the plot we also observe a special pattern: the time between subsequent GET requests is

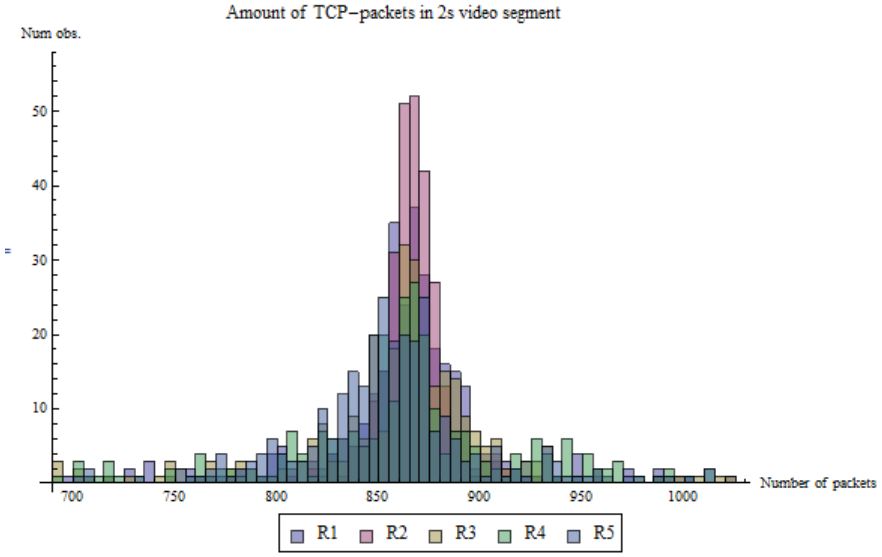

Figure 5.3: Number of packets within a 2s video segments encoded at 5 Mbps. The different colors are results from the various measurement runs.

not always 2 seconds. The variations we see in the figure seem to equalize each other as the average over the whole period is calculated to be ≈ 1.99 seconds. The same pattern is seen for all the measurement runs; see Table 5.1.

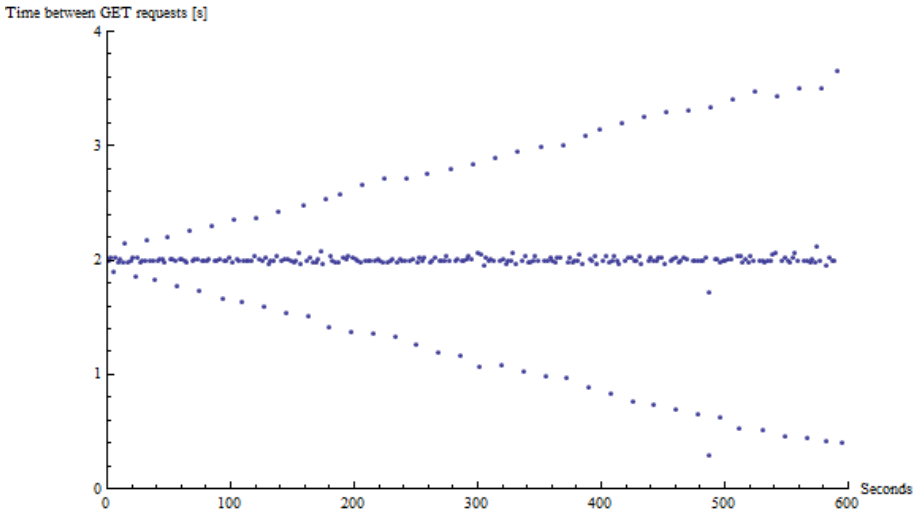

Figure 5.4: Plot illustrating the interarrival time for the various video GET requests. Tv2 Sumo, no loss.

	Mean Time Between GET request	Standard Deviation
min	1.9932	0.4870
max	2.0057	0.5060
avg	1.9994	-

Table 5.1: GET requests' interarrival time for the various measurement runs.

5.2.2 Comoyo

Available quality levels: {0.097, 0.197, 0.295, 0.493, 0.790, 1.177, 1.578, 2.178, 3.959, 5.934} Mbps.
Video: Varg Veum: I Mørket er alle ulver grå.

In this scenario we have observed that Comoyo throughout the various sessions plays video at the highest QL, which is 5.934Mbps. This was expected as our bandwidth limitation (still) is 10Mbps, meaning that streaming a video encoded at 5.934Mbps should be no problem. In Figure 5.5 we demonstrate this for one of the measurement runs.

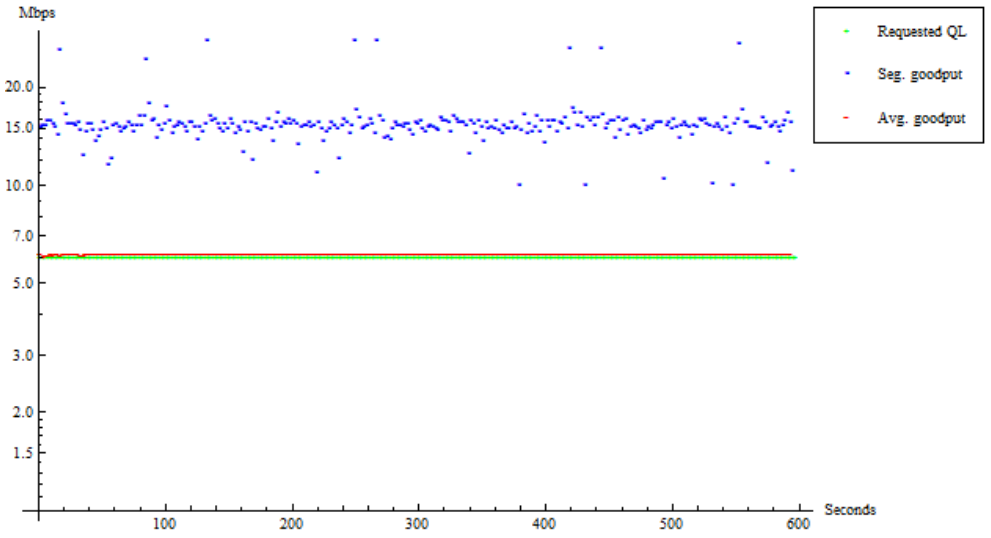

Figure 5.5: A plot showing segment goodput in the active period, average goodput and requested QL. Note the logarithmic y-axis. Comoyo, no packet loss.

To also get familiar with the content characteristics of Comoyo we have made a histogram showing the distribution of the various segment sizes (in terms of number of packets), shown in Figure 5.6 on the following page. We would expect the size to be $\frac{5934000bps*2s}{1460byte*8} \approx 1016$ packets, which corresponds well with what we observe from the figure.

Also in this scenario we started the Wireshark capture 1-2 minutes after having started the video playback. To check whether the player is in its steady state we have calculated the interarrival time for the various video GET requests. In Figure 5.7 on the next page we have plotted the results from one of the runs showing that the player sends a new video

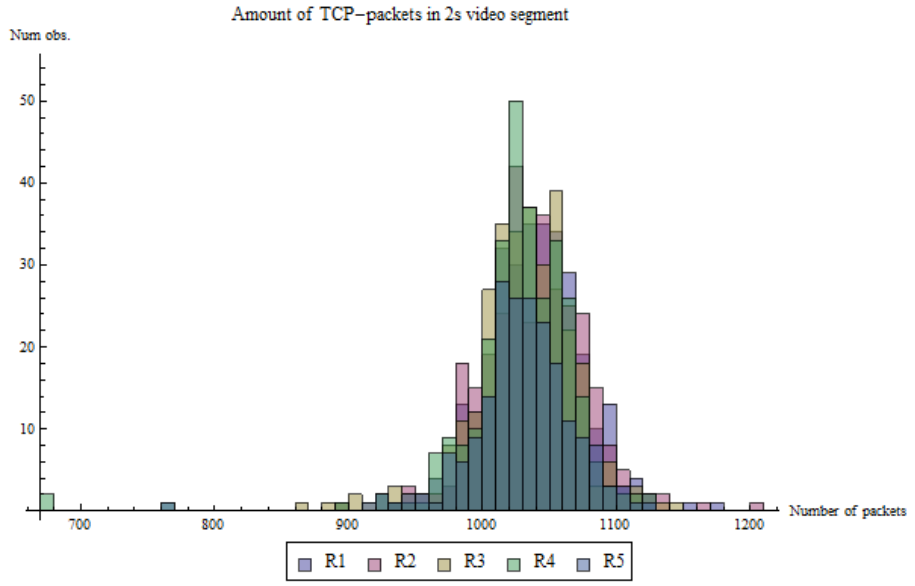

Figure 5.6: A histogram plot showing the number of packets registered within a video segment (at the highest QL) for each of the five runs.

GET every $\approx 2s$, implying that the player is in steady state. In contrast to Tv2 Sumo, we do not see the special pattern observed in Figure 5.4 on page 57; the player simply requests a new segment every ≈2 seconds. By looking at the relatively low standard deviation in Table 5.2 we see that this is the case for all runs.

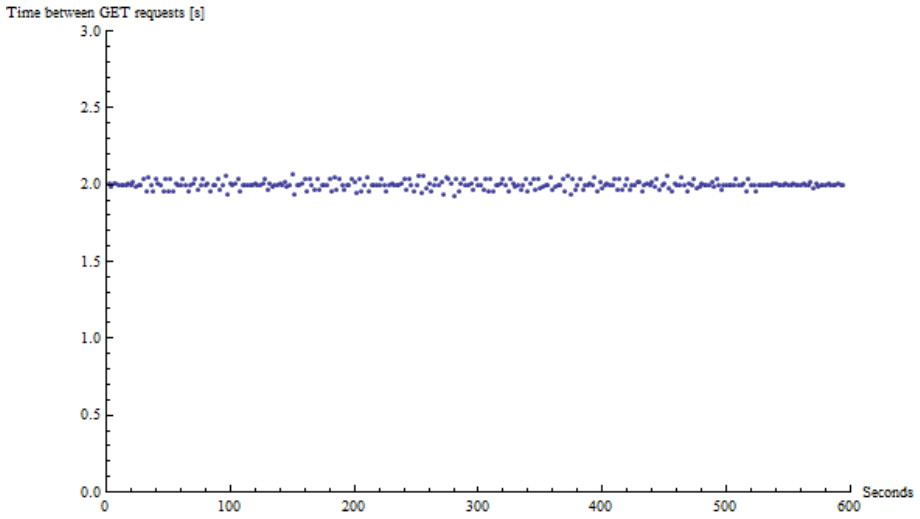

Figure 5.7: Time between video GET requests. Comoyo, no packet loss.

	Mean Time Between GET request	Standard Deviation
min	1.99991	0.0293
max	2.00021	0.0326
avg	2.00001	-

Table 5.2: Video GET requests' interarrival time for the various measurement runs.

5.3 Player's Reaction to the Introduced Packet Loss

In this section we look at the player's overall reaction to packet loss from the time when packet loss was introduced to the end of the Wireshark capture, see Figure 5.8.

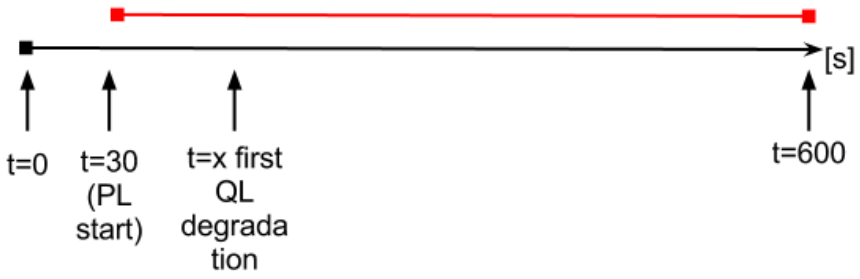

Figure 5.8: The red line denotes the time period which we are focusing on in this section.

The packet loss is, as explained in Section 4.2.1.3, introduced by the Click Modular router which looks at incoming TCP packets and drops them with a certain probability. In the proceding sections we look at the player's reaction at the 5- and 7% packet loss scenario, for both Tv2 Sumo and Comoyo. More specifically, we will focus on the quality metrics decribed and argued for in Section 3.5: quality transitions and the requested bitrate (quality level).

5.3.1 Tv2 Sumo

Available quality levels: {0.25, 0.75, 1.5, 2.5, 3.5, 5.0} Mbps.

5.3.1.1 5% Packet Loss

The player has reacted differently in all the different measurment runs. This was to be expected, as the entity dropping packets (Click) is a randomized process thus causing different behaviour at the network layer for each run. Figure 5.9 give an overview of what happens in both the network and in the player for one of the measurment runs.

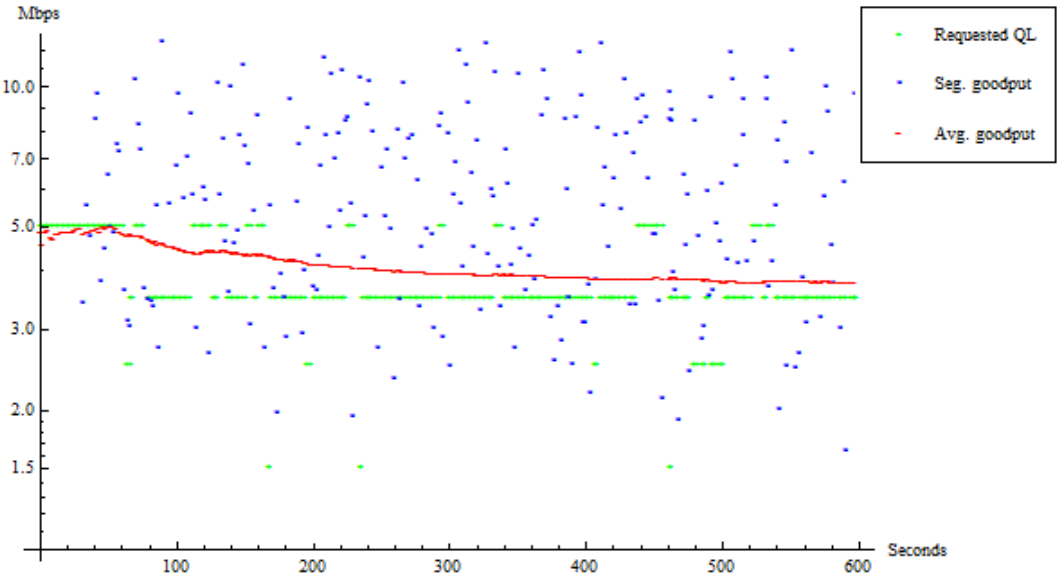

Figure 5.9: Plot illustrating the requested QL, the segment goodput in the active period and average goodput for Tv2 Sumo in the 5% packet loss scenario. Please note that the y-axis is logarithmic.

From the figure we can see that the player switches between different quality levels quite frequently (the green marks represent the requested QL). The player does not seem to stabilize at any level as we observe quality transitions throughout the session. What and for what reason quality transitions are triggered is part of the propietary rate adaptation logic inside the player, as described in Section 2.2.4.2. However, a natural apporoach for making such decisions is for the player to perform some available bandwidth estimation based on, for instance, the observed goodput. We have implemented and plotted the moving average (see the red line) of the various 2s video segments' goodput (the moving average technique is described in Section 5.1). The moving average can be regarded as an estimation of the available bandwidth as seen from the video player. For instance, in Figure 5.9, we see that when the

player for the first time decide to request video segments at a lower QL (at time = 64.47s) we can also see that the moving average (the red line) has started to go down. However, from the figure it is also clear that the video player do not use the moving average of the video segments's goodput for deciding whether to change QL as the player adjusts the requested QL up and down while the red line is quite steady (although going down). As the moving average takes the whole time period into account, short term changes in segment goodput will not be reflected as much as if a shorter averaging period were used. So if we were to guess on how the bandwidth estimation is done we would guess that a "sliding window" approach is used, taking the last X number of samples into account to better reflect what is currently happening, now, in the network. Implementing such a sliding window could be interesting, but is outside of the scope of this thesis. We will focus more on what we can observe.

	Number of QL transitions
Min	32
Max	36
Avg	34

Table 5.3: This table consists of max, min and the average number of quality transitions for the five different runs.

Number of QL transitions	QL transitions/minute	Avg. requested (QL) [Mbps]
32	3.37	3.08
36	3.81	3.26

Table 5.4: Table shows the number of QL transitions per minute and the average requested QL for the min and max values of the number of quality transition in a run for Tv2 Sumo 7% packet loss scenario.

Table 5.3 shows the that there is a high number of quality transistions for the different runs. In Table 5.4 the min and max values from Table 5.3 are connected to the average requested bitrate (QL) and the number of quality transitions per minute for the respective runs. From this we observe that the run with the lowest number of quality transitions (32) also has a lower average QL than the run with the highest number of quality transisitions. It could seem that the player strives to achieve the highest

QL at the cost of more quality transistions, as the player does not stabilize. Recall from Section 3.5 that this is could have a negative impact on the end user's QoE.

5.3.1.2 7% Packet Loss

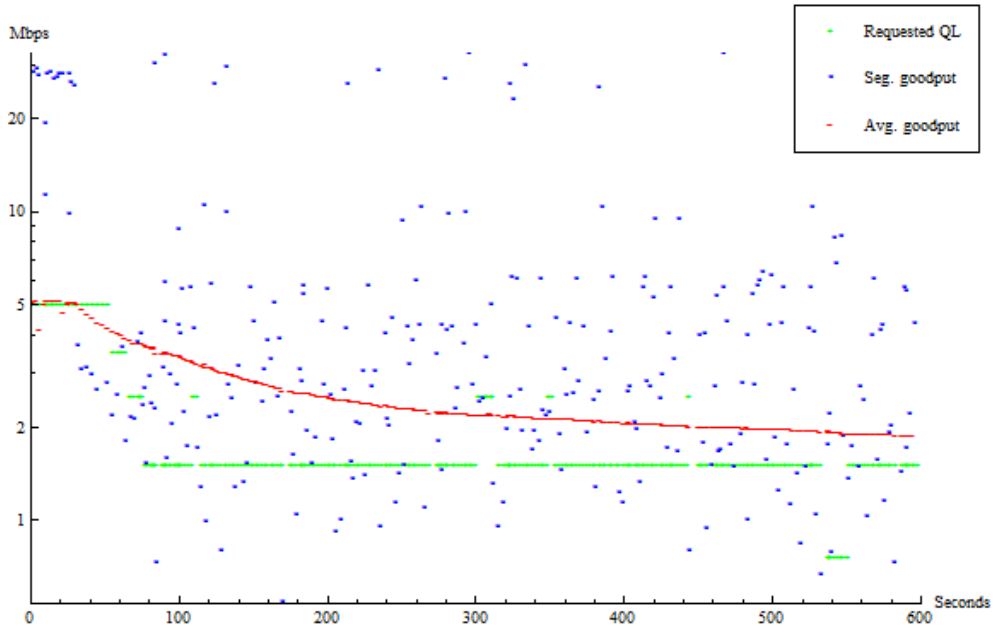

Figure 5.10: Plot illustrating the requested QL, the segment goodput in the active period and average goodput for Tv2 Sumo in the 7% packet loss scenario. Note the logarithmic y-axis.

	0% loss mean goodput [Mbps]	0% loss std. dev.	5% loss mean goodput [Mbps]	5% loss std. dev.	7% loss mean goodput [Mbps]	7% loss std. dev.
min	22.78	8.45	6.64	5.68	3.66	5.2
max	27.65	7.95	7.74	6.68	4.95	7.84
avg	25.26	-	7.22	-	4.23	-

Table 5.5: Mean goodput and the standard deviation for all scenarios.

Figure 5.10 shows one of the measurement runs for this scenario. The more packet loss that is introduced to the system, the longer time it will take to tranfer a certain amount of data. With a higher packet loss ratio

we will experience lower throughput and goodput, as we can see from Table 5.5. The average goodput (in the active period) from the 0% packet loss scenario is ≈ 25 Mbps, while it is ≈ 7 Mbps and ≈ 4 Mbps for the 5% and 7% loss scenarios, respectively. The introduced packet loss clearly impacts the system. In the previous section (in the 5% scenario) we saw that the average achieved QL was around 3 Mbps, which is much lower than the average 7 Mbps goodput. A reason for this could be the large variations in goodput (described in Table 5.5 by the high standard deviation), making it hard to estimate the available bandwidth. In the 7% packet loss scenario we therefore expect the achieved average QL to be lower than the average of the mean goodput (≈ 4Mbps) and lower than what was achieved in the 5% packet loss scenario. In terms of the number of quality transitions we have observed large variations in how the player has reacted; see Table 5.6.

	Number of QL transitions
Min	8
Max	25
Avg	15.8

Table 5.6: This table consists of max, min and the average number of quality transistion for the five different runs.

In Table 5.7 we can see from the runs with the min and max values of the number of quality transitions that the average requested QL is 1.52 and 1.23 Mbps, respectively. The requested QL has, as expected, decreased compared to the 5% packet loss scenario.

Number of QL transitions	QL transitions/minute	Avg. requested (QL) [Mbps]
8	0.85	1.52
25	2.64	1.23

Table 5.7: Shows the number of QL transitions per minute and the average requested QL for the min and max values of the number of quality transitions in a run for Tv2 Sumo's 7% packet loss scenario.

5.3.2 Comoyo

Available quality levels: {0.097, 0.197, 0.295, 0.493, 0.790, 1.177, 1.578, 2.178, 3.959, 5.934} Mbps

5.3.2.1 5% Packet Loss

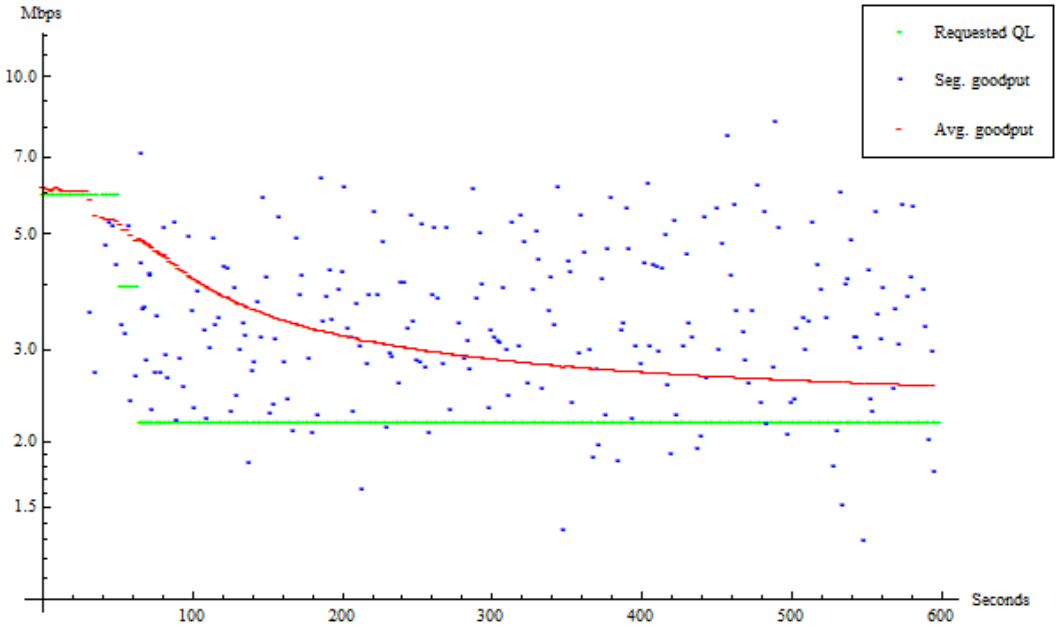

Figure 5.11: Plot illustrating the requested QL, the segment goodput in the active period and average goodput for Comoyo in the 5% packet loss scenario. Please note that the y-axis is logarithmic.

In this scenario the Comoyo player reacted differently from what we saw from Sumo in the preceeding sections. After the introduction of packet loss the player degrades the requested QL and stays at that level. Figure 5.11 demonstrates this behavior for one of the measurement runs. From the figure we can see that the player waits for some time after the introduction of packet loss to degrade the requested QL. The quality is degraded at time = 52.7s, 22.7s after the Click router started dropping incoming packets with 5% probability. In the beginning of the capture, before dropping packets, the red line (which shows the moving average of segment goodput, see Section 5.1) is as expected at the same level as the highest QL (5.934 Mbps). The introduced packet loss affects the average segment goodput (see Table 5.10) and from the red line in Figure 5.11 we can see that the segment goodput is decreasing. In the figure we see

that the quality degradation process is performed stepwise: the player goes from 5.934 Mbps to 3.959 Mbps before stabilizing at 2.178 Mbps. According to a study [24] (descibed in Section 3.3.1.2), end-users are favored with the insertion of intermediate quality levels before requesting the target QL, a behavior observed for all runs. Table 5.8 gives an overview for the various runs in terms of number of quality transitions. All runs have few quality transitions. In Table 5.9 the min and max values from Table 5.8 are shown alongside to their average requested QL, which shows an average of about 2 Mbps.

	Number of QL transitions
Min	2
Max	4
Avg	2.8

Table 5.8: This table consists of max, min and the average number of quality transistions for the five different runs in Comoyo's 5% packet loss scenario.

Number of QL transitions	QL transitions/minute	Avg. requested (QL) [Mbps]
2	0.21	2.15
4	0.42	2.12

Table 5.9: Table shows the number of QL transitions per minute and the average requested QL for the min and max values of the number of quality transitions in a run for Comoyo's 5% packet loss scenario.

	0% loss mean goodput [Mbps]	0% loss std. dev.	5% loss mean goodput [Mbps]	5% loss std. dev.	7% loss mean goodput [Mbps]	7% loss std. dev.
min	15.01	4.23	3.71	1.25	2.46	1.20
max	16.08	4.49	4.01	1.51	2.67	1.51
avg	15.58	-	3.86	-	2.53	-

Table 5.10: Mean goodput and the standard deviation for all Comoyo scenarios.

5.3.2.2 7% Packet Loss

In the 7% packet loss scenario Comoyo's player reacts differently from the 5% scenario. In this scenario we observe more quality transistions and a lower requested bitrate, Figure 5.12 demonstrates this. The lower requested bitrate was expected as the introduction of packet loss causes the mean segment goodput to decrease (see Table 5.10).

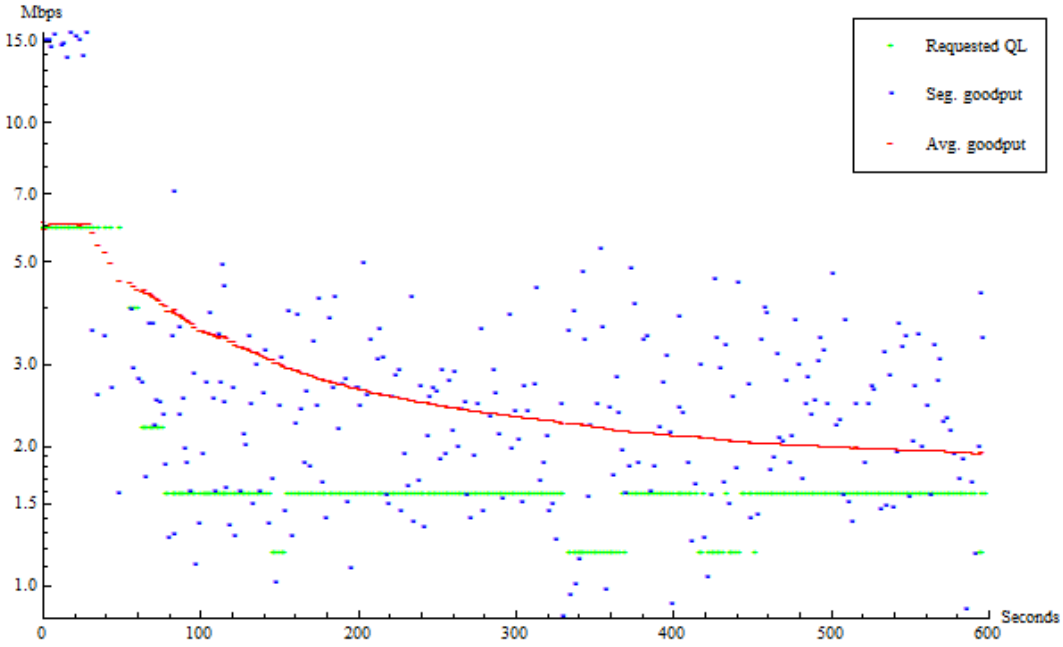

Figure 5.12: Plot illustrating the requested QL, the segment goodput in the active period and average goodput for Comoyo in the 7% packet loss scenario. Please note that the y-axis is logarithmic.

The average number of quality transitions for the five measurement runs is found to be 20, see Table 5.11. This is more than what we observed in Tv2 Sumo's case, where the average number of observed quality transistions for the 7% packet loss scenario was found to be 15.8. It was a bit suprising to see all these quality transitions in Comoyo's player, as we in the 5% packet loss scenario observed a smooth and stable player without many quality transitions. It is also hard to compare the two players as there are many factors that potentially influence the rate adaptation logic inside the players. This will be discussed in more detail in the next chapter.

	Number of QL transitions
Min	15
Max	26
Avg	20

Table 5.11: This table consists of max, min and the average number of quality transistions for the five different runs.

5.4 Time to the First Quality Level Degradation

One of the objectives in Section 4.1 was to find how long time it takes for the player to react to packet loss. In other words, we are interested in the time period shown in Figure 5.13. For all runs, packet loss is introduced 30 seconds after the Wireshark capture is started.

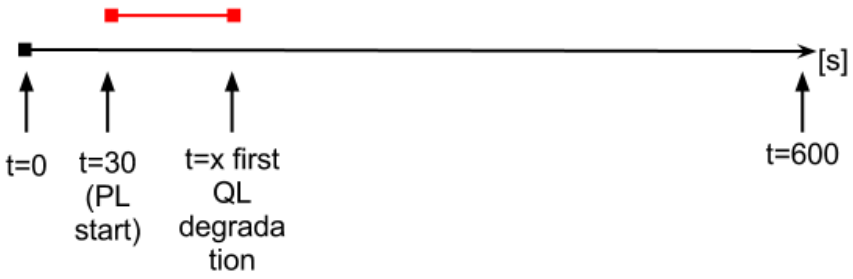

Figure 5.13: The red line denotes the time period which we are focusing on in this section.

5.4.1 Tv2 Sumo

5.4.1.1 5% Packet Loss

	Time For QL reduction [s]	Reaction Time [s]
min	40.18	10.18
max	64.47	34.47
avg	50.71	20.17

Table 5.12: Min, max and average values of the player reaction time to packet loss.

Table 5.12 gives an overview of the five runs where the time for QL reduction is the time in the capture for the QL reduction, while the reaction time is the time difference from the start of the packet loss until the first QL reduction. From the table we can observe that the reaction times vary quite a lot, by looking at the time difference of the min and max observations (34.47s-10.18s = 24.29s), which is huge. To further investigate if there is any connection between packet loss and the player reaction time we need to look more closely at the video trace files from the time when packet loss is introduced to the time when the QL degradation is registered.

What happens in the time period between the start of the packet loss and the time for the first QL degradation?

From Table 5.12 we see that the player's reaction time (time from packet loss start to QL degradation) varies quite a lot. As we are dealing with a random process (the Click router) we could expect different behavior for each run. To see if we can connect packet loss to the player's reaction time we look into the details of the various measurements. Earlier we have mentioned that there probably is an available bandwidth estimation algorithm helping the player to decide whether to request a new QL or not. We also have a hypothesis that there could be some buffer threshold triggering the player to reduce the QL. In this section we will investigate this and see if there could be such a buffer threshold.

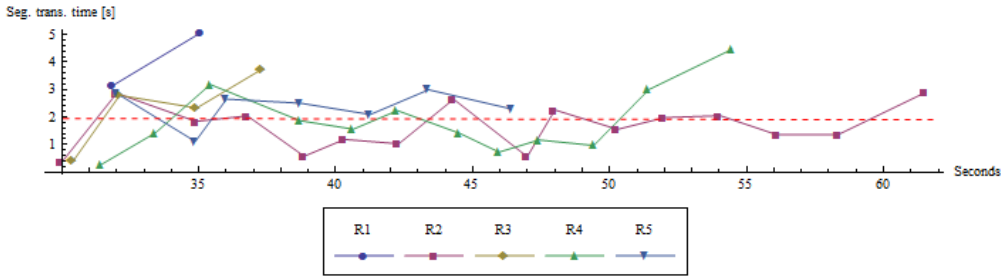

Figure 5.14: Plot illustrates the segment transmission time for all runs. Each point in the plot is {time for GET request, segment transmission time}; the last point denotes the last video segment transmitted before QL degradation.

We know that the player will have to use its playback buffer if a 2s video segment takes more than 2s to transfer. A video segment will naturally take longer time to transfer when packets within it are dropped.

When using TCP dropped packets are retransmitted, which itself introduces some extra delay as one has to take into account the extra transmission time and the time it takes for TCP to notice that a packet actually is lost and has to be resent. If the TCP version in use is a loss based one, we should expect the congestion window to be reduced, resulting in a lower throughput (see Section 2.3). A lower throughput means less information transferred per time unit, which will result in a longer segment transmission time. In Figure 5.14 on the preceding page we have plotted the segment transmission time for each video segment after the introduction of packet loss and before the QL degradation. A line is drawn between the points to better see which ones belong to the various runs. Each point in the figure is: {GET-request time, segment transfer time (time for 200 OK - time for GET request}. The last point for each run is the last video segment requested before the QL degradation. Points which are above the red horizontal line indicate that the buffer is utilized as they take more than 2 seconds to transfer. From the figure, it appears that the higher above this line the video segments in a run are, the shorter the time to QL degradation is. The blue line is an example of this, where only two video segments are requested before the quality is degraded and both segments lie relatively high above the red line.

From the figure it could look like there is some buffer threshold which triggers the quality degradation. To further investigate this we have estimated the buffer usage, as described in Section 5.1. In Figure 5.15 on the following page we have made a plot where each point show the buffer state at the time for the GET request, where the last point shows the buffer state for the time of requesting a video segment at a lower QL.

The red line in the figure is the mean value of the buffer size at the time for QL degradation, which we have calculated to be 6.83s. A QL degradation is triggered after using approximately 10s-6.83s = 3.17s of the buffer. As seen from Figure 5.15 on the next page, all the various runs lie around this value. So, if there is a buffer threshold triggering a QL degradation, the rate adaptation logic will be directly influenced by the segment transmission time. We therefore look at how packet loss affects the segment transmission time. This is information which perhaps could be of use for the player to make better rate adaptation decisions as it could reflect underlying TCP mechanisms to a greater extent.

Packet Loss and Segment Transmission Time

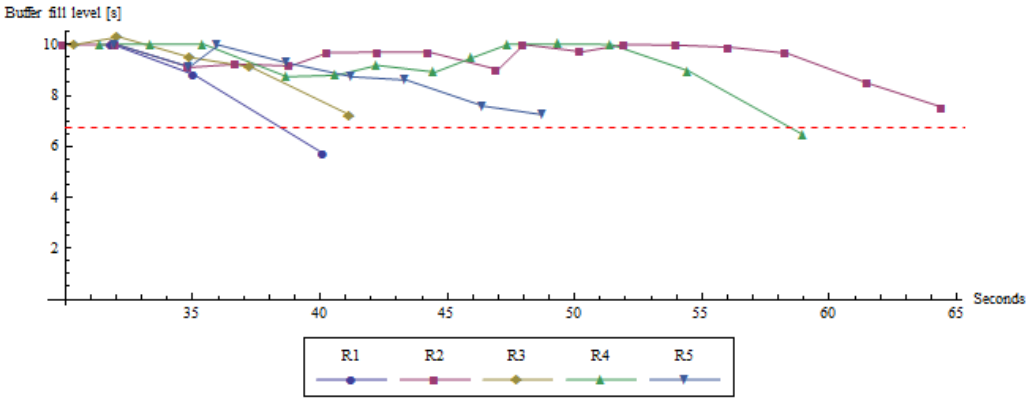

Figure 5.15: Plot illustrates the buffer fill level for the various runs of Tv2 Sumo 5% loss. Each point is the buffer level at the moment of a GET request. Note that the value 10s is just a reference value.

What we do know is that at time = 30s a process is started which drops incoming TCP packets with a certain probability. We do not know at what time the respective packets are dropped; however, from the video trace files we can find information whether a packet has been retransmitted. This can be done by looking at the info-field in the video trace file and look for packets flagged with "[TCP Retransmission]" or "[TCP Fast Retransmission]". This gives us more information about what is going on in the network as we can use these packets as indicators of packet loss. Since the Click router drops packets with a certain probability we expect the observed packet loss ratio (the number of retransmitted TCP video packets divided by the total number of TCP video packets) to be equal to the dropping probability.

Run #	Observed packet loss ratio
1	0.0402
2	0.0397
3	0.0402
4	0.0410
5	0.0414

Table 5.13: Observed packet loss ratio after packet loss introduction (time = 30s).

However, from Table 5.13 we can see that this is not the case. The observed packet loss ratio is about 0.04 for all runs, we expected it to be 0.05 for this scenario. For all scenarios we see this pattern: the observed

Scenario	Mean observed PL: Tv2 Sumo	Mean observed PL: Comoyo
5%	0.0405	0.0456
7%	0.0536	0.0641

Table 5.14: Mean observed packet loss for all scenarios.

packet loss ratio differs from the expected one, see Table 5.14. The reason for this is not known, and due to time limits we will not investigate it further. However, if this is to be done, we would suggest looking into:

- **Wireshark** (as we have experienced some problems with it) to check whether it really shows all packets which are restransmitted.
- **The TCP protocol**- are all retransmitted packets flagged as retransmitted?
- **The Click router configuration**- could Click be missing out on some packets?
- **The Cisco router** - bandwidth limitation, how does it impact packet loss?
- **The stochastic behavior** of the packet dropping process, will the observed loss ratio go towards the expected one for a longer measurement period?

There are many possible reasons for this behavior; however, we will continue our analysis using the information we have. Using "[TCP Retransmit]" and "[TCP Fast Retransmit]" as packet loss indicators. To visualize how packet loss influences the segment transmission time we have made a plot showing this; see Figure 5.16 on the following page. The plot consists of data from all the five runs (for the 5% scenario) where the segment transmission time is plotted together with the amount of lost packets within the respective video segment.

We can see a tendency that the more packets that are lost, the longer the transmission time is. However, we also observe that video segment with about the same amount of packets lost can have very different transmission times. Underlying TCP mechanisms can cause the throughput

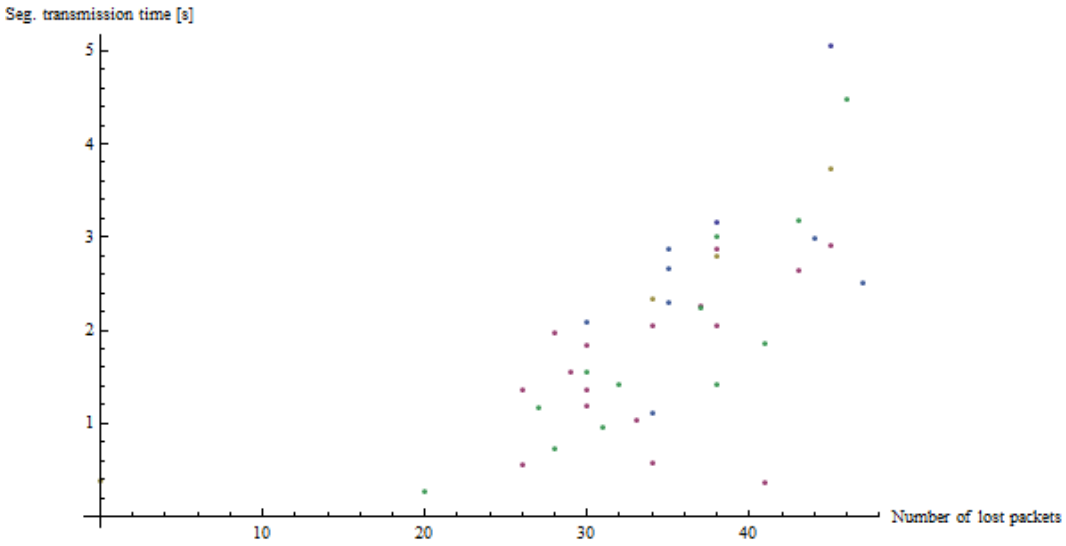

Figure 5.16: XY plot showing the segment transmission time and the number of lost packets within the respective segment.

to vary, thus influencing the segment transmission time. This is probably the reason that we observe different transmission times for the same amount of packet loss.

5.4.1.2 7% Packet Loss

In the 7% loss scenario we would expect the player to react faster than for the 5% scenario, as more packets lost lead to lower goodput and longer segment transmission times, thus reaching the buffer threshold faster. Table 5.15 gives an indication that the player do react a bit faster than for the 5% scenario. However, the difference is not big at all: the average reaction time in the 5% scenario is 20.17 seconds, while in the 7% scenario it is 17.75 seconds.

	Time For QL reduction [s]	Reaction Time [s]
min	36.27	6.27
max	59.54	29.54
avg	47.75	17.75

Table 5.15: Min, max and average values of the player reaction time to packet loss in the 7% loss scenario.

Like in the 5% scenario, we have estimated the buffer usage to see if we can see the same buffer threshold. Figure 5.17 on the next page,

however, shows that this is not the case. The buffer threshold which we thought we had found for the 5% scenario can not be seen in this figure. The mean buffer utilization is found to be $\approx 7.65s$ (10s - mean buffer level). The mean buffer level at the time for QL degradation is shown in Figure 5.17 as the red line (2.35s). However, as we can see from the figure, the various runs all degrades the quality having utilized very different amounts of the buffer. So from this we cannot connect the player's reaction time to buffer utilization and segment transmission time. What we can say about the player's reaction time to the introduced packet loss is just what we have observed and summarized in Table 5.15.

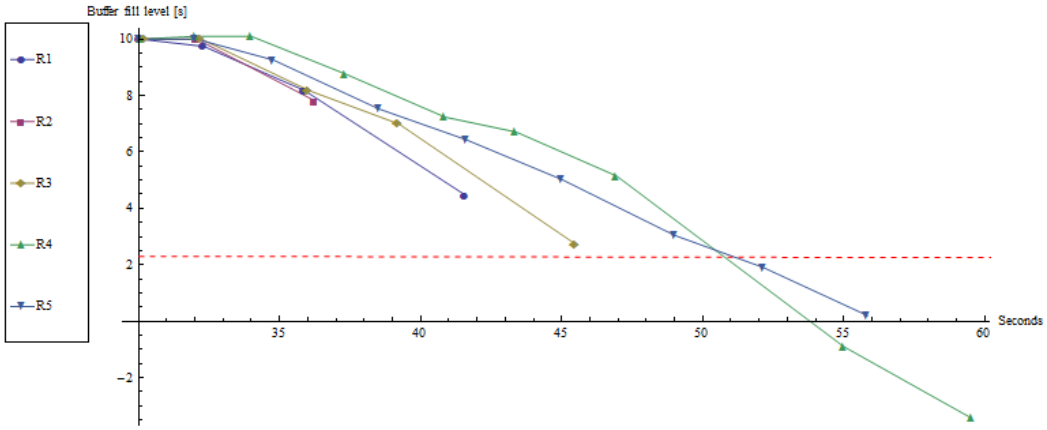

Figure 5.17: Plot illustrates the buffer fill level for the various runs for the 7% packet loss for Tv2 Sumo. Each point is the buffer level at the moment of a GET request. Note that the value 10s is just a reference value.

5.4.2 Comoyo

5.4.2.1 5- and 7% Packet Loss

As the results for the 5- and 7% packet loss scenario in Comoyo's case do not differ much, we have merged them together in this section.

When looking at the time between the packet loss introduction and the time for the first QL degradation we, had similar expectations as for what we have seen in Tv2 Sumo: large variations in the player's reaction time. For the 5% packet loss scenario we do observe variations in reaction time (see Table 5.16), although not as big ones as observed for Tv2 Sumo.

	Time For QL reduction [s]	Reaction Time [s]
min	43.91	13.91
max	52.70	22.70
avg	47.83	17.83

Table 5.16: Min, max and average values of the player reaction time to packet loss. Comoyo, 5% packet loss.

As seen from Table 5.17 the results are not very different for the 7% packet loss scenario.

	Time For QL reduction [s]	Reaction Time [s]
min	49.02	19.02
max	56.15	26.15
avg	52.37	22.37

Table 5.17: Min, max and average values of the player reaction time to packet loss. Comoyo, 7% packet loss.

We also estimated the buffer utilization to see if there was any buffer threshold triggering the QL degradation. The pattern we saw in Section 5.4.1.1 was, however, not found in the two packet loss scenarios for Comoyo's player. From the measurements we performed it does not seem that the QL degradation is being triggered by a buffer threshold. For the interested reader we have added two figures showing the buffer utilization for Comoyo's 5- and 7% packet loss scenario in Appendix C. What we can say about the player's reaction time to packet loss is what is shown in Table 5.16 and Table 5.17. The average player reaction time is $\approx 20s$.

Chapter 6

Discussion and Evaluation

OTT service providers are, as previously described, completely dependent on the underlying best-effort network. In order provide the best service possible there are many important aspects to consider. However, the focus of this thesis has been on how packet loss in the network (QoS) affects the video player and thus the end user's QoE. The players we have performed measurements on have during the various measurement runs shown that they are indeed affected adversly by packet loss.

The packet loss which is introduced by a Click Modular Router (see Section 4.2.1.3) has various implications on the network performance and thus on the video player's performance. Table 6.1 shows the mean segment goodput for all the different scenarios.

	Tv2 Sumo mean goodput[Mbps]	Comoyo mean goodput[Mbps]
0% loss	25.26	15.58
5% loss	7.22	3.86
7% loss	4.23	2.53

Table 6.1: Mean goodput for all scenarios for Comoyo and Tv2 Sumo.

First, we want to stress that Table 6.1 shows the mean segment goodput from the active period (see definition in Section 5.1); this is why we in the 0% packet loss scenario observe goodput values above the 10 Mbps bandwidth limitation. For the 0% loss scenario we would expect the mean segment goodput to be similar for Comoyo and Tv2 Sumo. As seen from the table, this is not the case. A possible explanation for the

observed differences is the underlying TCP protocol, which can cause differences in throughput. We can observe from the differences in goodput on the various loss scenarios that packet loss clearly has an impact on the network performance. Now, the question now is how this influences the players' performance, which is interesting as it has been shown that application performance metrics do impact end users' QoE. This brings us back to the main objectives for performing the conducted measurements. In Section 4.1 we defined two main objectives which we will discuss in Section 6.1 and Section 6.2.

6.1 The Player's Reaction to Packet Loss

Objective one: Will the player stabilize at some quality level, or will we see many quality transitions throughout the session, after the introduction of packet loss?

In Section 5.3 we presented results for the "Player's reaction to the introduced packet loss" addressed the question posed in objective one. In terms of quality transitions we have found that in all packet loss scenarios, expect for one (Comoyo, 5% loss), the player does *not* seem to stabilize; Quality transitions were observed throughout the various sessions. Quality transitions were most frequently observed in Tv2 Sumo's 5% packet loss scenario. For the measurement run with the most quality transitions, we observed 3.81 QL transitions per minute, and at the same time achieved a slightly higher bitrate than for the other runs. It could seem that the player strives to achieve the highest QL at the cost of more quality transitions. A player which always strives for the highest QL, not taking the expense of eventual quality transitions into account, could have a negative impact on the end users' QoE (recall Section 3.5). In Figure 5.9 on page 62 (from Tv2 Sumo, 5% loss) we saw examples of many short-term quality changes which could potentially harm the QoE, even if a higher bitrate is achieved. These short-term changes in quality are observed at several points in the 10 minute long measurement period, also close to the end. Such striving for the highest QL is a behavior also found in the Netflix player [1].

Comoyo's behavior in the 5% packet loss scenario seems to be more conservative. From the results we see that the player performs a stepwise quality degradation (as recommended in [24]) and stays at the target quality level throughout the various sessions, which in this case is

2.178 Mbps. Could it be that the rate adaptation logic for this player is different? Our results from the 7% packet loss scenario indicate that this is not the case. The player's behavior in the 7% scenario is not as smooth; we have observed unstable sessions with many quality transitions. Why do we see a different behavior in the 5% scenario? We do not know; however, we have a theory that it may be caused by the content characteristics. By looking at the quality levels of the content we have performed measurements on in Comoyo's case ({0.097, 0.197, 0.295, 0.493, 0.790, 1.177, 1.578, 2.178, 3.959, 5.934} Mbps) we see that there is a large gap between the 2.178 Mbps and 3.959 Mbps quality levels. It could be that the player wants to upgrade the quality, but that the next quality level is too far away from the player's estimate of the current bandwidth.

Thus, from what we have observed it appears that neither player takes the cost of quality transitions as it may harm end users' QoE into account in their rate adaptation logic. If they had so we would expect the players to stabilize at some quality level, which we cannot see from our results (except for the already commented Comoyo 5% loss scenario). However, one can question how realistic our testing scenario is: random packet loss over an extended time period. Does the players really need to handle such a scenario? Regardless of the how realistic the scenario is, we believe that the player should be able to handle the observed effects of it: frequent quality transitions. It should not be difficult to implement some logic into the player saying something along the lines of "if the player has experienced X number of quality transitions in the last couple of minutes, then stabilize at some lower quality level." The difficult part would be answering questions like: how long time do one need to stream video at a higher quality level for it to be worth the cost of having to degrade the the quality? This is difficult to answer without performing subjective tests. What we know from what has been done in this area is that users are favored (higher QoE) with fewer quality transitions at the expense of a higher bitrate. Evaluating the quality of adaptive video streams using quality transitions as a quality metric in relation to bitrate is therefore difficult as we do not know their exact relationship. We would like, however, to stress the importance of quality transitions as quality metric as it do have an impact on the perceived quality, but might at the same time be "forgotten" in the players' strive for reaching the highest QL.

6.2 The Player's Reaction Time

Objective two: How much time will pass before the player triggers its first quality transition?

In Section 5.4 we presented the results that address objective two. We see that the time it takes, for the player to react to the introduced packet loss by triggering a QL degradation varies significantly for all scenarios. The measurement run with the fastest reaction time (from Tv2 Sumo, 7% loss) used 6.27s before the player triggered a QL degradation, whereas the reaction time of the run with the longest reaction time was observed to be 34.47s (Tv2 Sumo, 5% loss). Comoyo's results all lie in between these values, and we observe variations in this player's reaction time as well.

We suspect the varying results to be consequence of the randomized behavior of the packet dropping process. Due to its random behavior we do not expose the players for the same amount of packet loss each time, and we should therefore expect different behavior for each measurement run. For this reason we performed further investigation on the players' behavior in the time interval between the introduction of packet loss and the time for the first QL degradation. We wanted to see if there was a strong connection between packet loss and the player's decision to degrade the quality. As packet loss affects a connection's throughput thus causing less bits to be transferred per time unit, we also know that the transmission time of a 2s video segment would be longer. From this we made a hypothesis that there could be a buffer threshold triggering the player to degrade the requested QL. If a 2s video segment takes more than 2s to transfer; the playback buffer will have to be utilized. We therefore investigated the 2s video segments' transmission times within each measurement run for the specified time interval.

Our results from Tv2 Sumo, in the 5% packet loss scenario, indicated that there could be such a buffer threshold. Our calculations showed that in all measurement runs for this scenario a QL degradation was triggered after using approximately 3s of the playback buffer. In the investigation process it was noted that some of the 2s video segements had very different transmission times. Furthermore, we wanted to check if this could be related to the introduced packet loss. In Figure 5.16 on page 74 we saw indications for such a connection where the trend says that the more packets lost within a video segment, the longer the segment transmis-

sion time will be. However, from this figure we can also observe large variations where segments with approximately the same amount of lost packets have very different transmssion times. This behaviour may be caused by TCP mechanisms. So, *if* there is a buffer threshold, we might have found some connections that link packet loss to segment transmission time, which in turn is essential for the buffer utilization and hence what makes the player trigger a QL degradation.

However, this buffer threshold was only found for Tv2 Sumo's 5% packet loss scenario. In the other scenarios the variations of buffer the utilization at the time of the first QL degradation are high and we don't see any indications that there is a buffer threshold.

To answer objective two: we cannot say more than that we have observed player reaction times between approximately 6 and 35 seconds, and that the players' reaction time in all measurement runs vary a lot. However, if we were to make a recommendation for the desired behavior we would like to see more of the behavior of the Netflix player observed in [1]. Netflix' player has a large playback buffer which it utilizes under fluctuating network conditions rather than degrading the quality. In this way one can avoid unnecessary quality transitions under short-term network fluctuations. This would be at the cost of a longer initial delay for accumulating the large playback buffer. However, as mentioned previously in [22] it was found that users are generally willing to tolerate a longer start-up delay for a better viewing experience - thus it is probably worth the cost. To sum up: we would like to see the players utilize more data from its buffer before degrading the quality, as this might reduce the number of quality transitions under short-term network fluctuations.

Chapter 7

Conclusion

In this thesis we have followed the methodology described in Section 1.2. A literature study was first conducted where we reviewed state of the art on QoS and QoE. The outcome of this study was the proposal of several quality metrics applicable to adaptive video streaming, amongst them: *initial buffering time*, *mean duration of a rebuffering event*, *rebuffering frequency*, *quality transitions* and *bitrate*. A higher bitrate will in most cases lead to a higher degree of QoE, however, according to other research users might prefer a stable video stream with few quality transitions at the cost of an overall higher bitrate. Through measurements on the players of Tv2 Sumo and Comoyo we wanted to see if this is taken into consideration by today's adaptive video streaming technologies.

The results discussed in Section 6.1 imply that, for our measurement scenarios, neither player takes quality transitions into account in their rate adaptation logic. For both players a relatively large number of quality transitions have been observed throughout the various sessions, implying that they do not stabilize, but rather strive for a higher quality level. If we were to give any recommendations to the OTT service providers we would advise them to investigate the effects of quality transitions and consider including a solution for handling potentially negative effects (in terms of QoE) in the rate adaptation logic of the video player.

7.1 Future Work

Much of the work of this thesis is based upon research that has found that users prefer a stable video stream with few quality transitions at the expense of an overall higher bitrate. Our focus has mainly been on the relationship between quality transitions and the bitrate. However, as discussed in Section 6.1, we are not sure of the exact relationship between the two quality metrics; we just know that keeping a low number of quality transitions is desirable. Quantifying the relationship between the metrics would further enhanced the thesis as we could possible have done a better estimate the end users' QoE. This could, for instance, be done through subjective testing.

Performing the measurements on other technologies than Microsoft's Smooth Streaming would be interesting and give an even broader perspective on how today's technologies on adaptive video streaming works.

An area we slightly touched upon in Section 5.4.1.1 was TCP's influence on segment transmission times. Further investigation of this, in order to see how the adaptive TCP works together (or against?) the adaptive video player, would be interesting.

Bibliography

[1] S. AKHSHABI, A. C. BEGEN, AND C. DOVROLIS, *An experimental evaluation of rate-adaptation algorithms in adaptive streaming over http*, in Proceedings of the second annual ACM conference on Multimedia systems, MMSys '11, New York, NY, USA, 2011, ACM, pp. 157–168. [cited at p. 50, 78, 81]

[2] A. BEGEN, T. AKGUL, AND M. BAUGHER, *Watching Video over the Web: Part 1: Streaming Protocols*, Internet Computing, IEEE, 15 (2011), pp. 54 – 63. [cited at p. 8, 9, 11, 12, 13]

[3] P. BROOKS AND B. HESTNES, *User measures of quality of experience: why being objective and quantitative is important*, Network, IEEE, 24 (2010), pp. 8 –13. [cited at p. 23, 27, 31, 32, 33]

[4] CISCO, *Cisco Visual Networking Index: Forecast and Methodology, 2010-2015*. Whitepaper, June 2011. [cited at p. 1, 3]

[5] ——, *Cisco Visual Networking Index: Global Mobile Data Traffic Forecast Update, 2011-2016*. Whitepaper, February 2012. [cited at p. 6]

[6] W. COOOPER AND G. LOVELACE, *IPTV guide*, in IPTV Conference 2007 - Deployment and Service Delivery, IET, dec 2007, pp. 1 –66. [cited at p. xvi, 3, 4, 5, 6]

[7] N. CRANLEY, P. PERRY, AND L. MURPHY, *User perception of adapting video quality*, Int. J. Hum.-Comput. Stud., 64 (2006), pp. 637–647. [cited at p. 31, 35]

[8] K. DE MOOR, I. KETYKO, W. JOSEPH, T. DERYCKERE, L. DE MAREZ, L. MARTENS, AND G. VERLEYE, *Proposed framework for evaluating quality of experience in a mobile, testbed-oriented living lab setting*, Mobile Networks and Applications, 15 (2010), pp. 378–391. [cited at p. 23, 24, 25, 26, 27]

[9] M. EL-GENDY, A. BOSE, AND K. SHIN, *Evolution of the Internet QoS and support for soft real-time applications*, Proceedings of the IEEE, 91 (2003), pp. 1086 – 1104. [cited at p. 21, 22]

[10] M. FIEDLER, K. KILKKI, AND P. REICHL, *From Quality of Service to Quality of Experience*, may 2009. Dagstuhl Seminar. [cited at p. 23]

[11] ITU-T, *ITU-T Recommendation E.800: Terms and defnitions related to quality of service and network performance including dependability*, 1994. [cited at p. 23]

[12] ——, *ITU-T P.800. Methods for Subjective Determination of Transmission Quality - Series P: Telephone Transmission Quality; Methods for Objective and Subjective Assessment of Quality*, 1996. [cited at p. 26]

[13] ——, *ITU-T Recommendation P.10/G.100: Amendment 2: New Definitions for Inclusion in Recommendation ITU-T P.10/G.100*, 2008. [cited at p. 24]

[14] A. KIST, *A framework to evaluate performance from an application and user perspective*, in Australasian Telecommunication Networks and Applications Conference (ATNAC), 2011, nov. 2011, pp. 1 –6. [cited at p. 22, 32]

[15] E. KOHLER, R. MORRIS, B. CHEN, J. JANNOTTI, AND M. F. KAASHOEK, *The click modular router.*, ACM Trans. Comput. Syst., 18 (2000), pp. 263–297. [cited at p. 40]

[16] J. F. KUROSE AND K. W. ROSS, *Computer Networking: A Top-Down Approach (4th Edition)*, Addison Wesley, 4 ed., 2007. [cited at p. 8]

[17] R. KUSCHNIG, I. KOFLER, AND H. HELLWAGNER, *An evaluation of TCP-based rate-control algorithms for adaptive internet streaming of H.264/SVC*, in Proceedings of the first annual ACM SIGMM conference on Multimedia systems, MMSys '10, New York, NY, USA, 2010, ACM, pp. 157–168. [cited at p. 10]

[18] K. LAGHARI AND K. CONNELLY, *Toward total quality of experience: A QoE model in a communication ecosystem*, Communications Magazine, IEEE, 50 (2012), pp. 58 –65. [cited at p. 24]

[19] M. LERVOLD, *Measuring perceptual quality in internet television*, master's thesis, NTNU, 2009. [cited at p. 4, 23]

[20] MICROSOFT, *IIS Smooth Streaming Client Manifest*. http://msdn.microsoft.com/en-us/library/ee673438(v=vs.90).aspx. Accessed: 12/05/2012. [cited at p. xviii, 17]

[21] ——, *What is Silverlight*. http://www.microsoft.com/silverlight/what-is-silverlight/. Accessed: 11/05/2012. [cited at p. 16]

[22] R. MOK, E. CHAN, AND R. CHANG, *Measuring the quality of experience of HTTP video streaming*, in Integrated Network Management (IM), 2011 IFIP/IEEE International Symposium on, may 2011, pp. 485 –492. [cited at p. 29, 30, 34, 81]

[23] R. K. MOK, E. W. CHAN, X. LUO, AND R. K. CHANG, *Inferring the QoE of HTTP video streaming from user-viewing activities*, in Proceedings of the first ACM SIGCOMM workshop on Measurements up the stack, W-MUST '11, New York, NY, USA, 2011, ACM, pp. 31–36. [cited at p. 27, 31]

[24] R. K. P. MOK, X. LUO, E. W. W. CHAN, AND R. K. C. CHANG, *QDASH: a QoE-aware DASH system*, in Proceedings of the 3rd Multimedia Systems Conference, MMSys '12, New York, NY, USA, 2012, ACM, pp. 11–22. [cited at p. 30, 31, 35, 67, 78]

[25] NOKIA, *Quality of Experience (QoE) of Mobile Services: Can It Be Measured and Improved?* Whitepaper, 2004. Finland. [cited at p. 31, 32]

[26] A. OODAN, K. WARD, C. SAVOLAINE, M. DANESHMAND, AND P. HOATH, *Telecommunications Quality of Service Management: From Legacy to Emerging Services*, IEE Telecommunications Series, Institution of Electrical Engineers, 2003. [cited at p. 9, 32]

[27] T. PORTER AND X.-H. PENG, *An Objective Approach to Measuring Video Playback Quality in Lossy Networks using TCP*, Communications Letters, IEEE, 15 (2011), pp. 76 –78. [cited at p. 29, 30]

[28] J. SHAIKH, M. FIEDLER, AND D. COLLANGE, *Quality of Experience from user and network perspectives*, Annals of Telecommunications, 65 (2010), pp. 47–57. 10.1007/s12243-009-0142-x. [cited at p. 28, 31]

[29] I. SODAGAR, *The MPEG-DASH Standard for Multimedia Streaming Over the Internet*, IEEE MultiMedia, 18 (2011), pp. 62–67. [cited at p. xiii, 13, 14, 15]

[30] STREAMINGMEDIA, *Comment from netflix engineer*. http://www.streamingmedia.com/Articles/ReadArticle.aspx?ArticleID=79409. Accessed: 02/06/2012. [cited at p. 14]

[31] A. TANENBAUM, *Computer Networks*, Prentice Hall Professional Technical Reference, 4th ed., 2002. [cited at p. 18, 19]

[32] B. J. VILLA AND P. E. HEEGAARD, *Improving Perceived Fairness and QoE for Adaptive Video Streams*, The Eighth International Conference on Networking and Services (ICNS 2012), St. Maarten, Netherlands Antilles, march 2012. [cited at p. 30]

[33] B. WANG, J. KUROSE, P. SHENOY, AND D. TOWSLEY, *Multimedia streaming via TCP: an analytic performance study*, in Proceedings of the 12th annual ACM international conference on Multimedia, MULTIMEDIA '04, New York, NY, USA, 2004, ACM, pp. 908–915. [cited at p. 10]

[34] S. WINKLER, *Video Quality and Beyond*, in Proceedings of the 15th European Signal Processing Conference, Sept. 2007. [cited at p. 25]

[35] S. WINKLER AND P. MOHANDAS, *The Evolution of Video Quality Measurement: From PSNR to Hybrid Metrics*, Broadcasting, IEEE Transactions on, 54 (2008), pp. 660 –668. [cited at p. 24, 25, 26, 27, 28, 29]

[36] WIRESHARK, *About wireshark*. http://www.wireshark.org/about.html. Accessed: 02/06/2012. [cited at p. 89]

[37] A. ZAMBELLI, *IIS Smooth Streaming Technical Overview*. Whitepaper, Microsoft Corporation, March 2009. [cited at p. 8, 9, 10, 11, 12, 16, 17, 18]

[38] W. ZHAO, D. OLSHEFSKI, AND H. SCHULZRINNE, *Internet Quality of Service: an Overview*, tech. report, Columbia University, feb. 2000. [cited at p. 22]

Appendix A

Wireshark

Wireshark is the world's foremost network protocol analyzer, and is the de facto standard across many educational institutions and industries [36].

Figure A.1 shows a screenshot of a Wireshark capture. We have used Wireshark to capture and analyse video traffic sent across the network interface. This has been of great value to quantify a video player's performance.

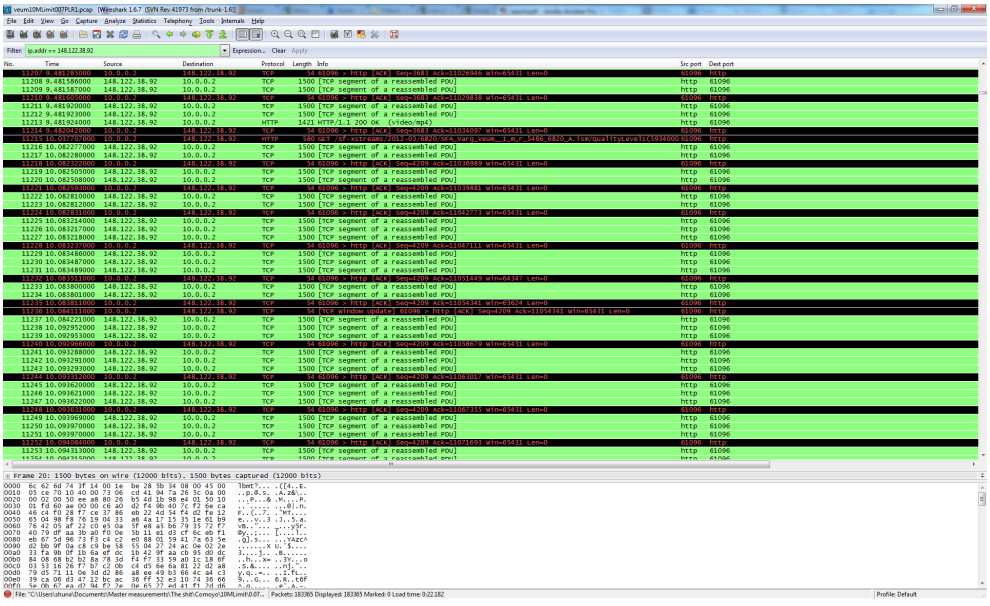

Figure A.1: A screenshot of Wireshark while capturing video traffic.

Appendix B

Click Configuration

Listing B.1: Drop incoming TCP packets with a certain probability.

```
b :: Counter;
c :: Counter;
d :: Counter;
b1 :: Counter;
c1 :: Counter;
d1 :: Counter;

// Shared IP input path and routing table
ip :: Strip(14)
    -> CheckIPHeader(INTERFACES
       129.241.200.119/255.255.255.0
       129.241.197.105/255.255.255.248)
    -> rt :: StaticIPLookup(
  129.241.200.119/32 0,
  129.241.200.255/32 0,
  129.241.200.0/32 0,
  129.241.197.105/32 0,
  129.241.197.111/32 0,
  129.241.197.104/32 0,
  129.241.200.0/255.255.255.0 1,
  129.241.197.104/255.255.255.248 2,
  255.255.255.255/32 0.0.0.0 0,
  0.0.0.0/32 0,
  0.0.0.0/0.0.0.0 129.241.200.1 1);
```

```
// ARP responses are copied to each ARPQuerier and the
   host.
arpt :: Tee(3);

// Egne queues

bv_ip_q1 :: ThreadSafeQueue(100);
bv_ip_q2 :: ThreadSafeQueue(100);

// Input and output paths for eth1

c_eth1 :: Classifier(12/0806 20/0001, 12/0806 20/0002,
   12/0800, -);
out_eth1 :: Queue(100) -> to_Eth1 :: ToDevice(eth1);
arpQ_eth1 :: ARPQuerier(129.241.200.119, 00:19:b9
   :13:61:45) -> out_eth1;

find_tcp_ack :: IPClassifier(ip proto 1, -);
find_data_eth2 :: IPClassifier(tcp, -);

FromDevice(eth1) -> c_eth1;
c_eth1[0] -> arpResp_eth1 :: ARPResponder(129.241.200.119
   00:19:b9:13:61:45) -> out_eth1;
c_eth1[1] -> arpt;
c_eth1[2] -> ip;
c_eth1[3] -> Print("eth1 non-IP") -> Discard;

arpt[0] -> [1]arpQ_eth1;

// Forwarding path for eth1
rt[1] -> DropBroadcasts
      -> b
      -> find_tcp_ack
      -> c
      -> RandomSample(DROP 0)
      -> d
      -> bv_ip_q1
      -> DelayUnqueue(0)
      -> [0]arpQ_eth1;

find_tcp_ack[1] -> [0]arpQ_eth1;
```

```
// Input and output paths for eth2
c_eth2 :: Classifier(12/0806 20/0001, 12/0806 20/0002,
   12/0800, -);
out_eth2 :: Queue(100) -> to_Eth2 :: ToDevice(eth2);
arpQ_eth2 :: ARPQuerier(129.241.197.105, 00:04:76:8f:a9:e5
   ) -> out_eth2;

FromDevice(eth2) -> c_eth2;
c_eth2[0] -> arpResp_eth2 :: ARPResponder(129.241.197.105
   00:04:76:8f:a9:e5) -> out_eth2;
c_eth2[1] -> arpt;
c_eth2[2] -> ip;
c_eth2[3] -> Print("eth2 non-IP") -> Discard;

arpt[1] -> [1]arpQ_eth2;

// Forwarding path for eth2
rt[2] -> DropBroadcasts
      -> b1
      -> find_data_eth2
      -> c1

// ********************************************************
// ************* ENDRE TALLET HER
   ************************

      -> RandomSample(DROP 0.07)

// ********************************************************
// ********************************************************

      -> d1
      -> bv_ip_q2
      -> DelayUnqueue(0)
      -> [0]arpQ_eth2;

find_data_eth2[1] -> [0]arpQ_eth2;

// Local delivery
toh :: ToHost;
```

```
ping_ipc :: IPClassifier(icmp type echo, -);
arpt[2] -> toh;
rt[0]   -> IPReassembler
        -> ping_ipc;
ping_ipc[0] -> ICMPPingResponder -> [0]rt;
ping_ipc[1] -> EtherEncap(0x0800, 1:1:1:1:1:1,
   2:2:2:2:2:2) -> toh;

s :: Script(
            label begin_loop,
            wait 10000ms,
            print "Incoming packets (All/TCP/TCP after
               drop): " $(b1.count) $(c1.count) $(d1.count
               ),

            // write d1.reset,
            // write b1.reset,
            // write c1.reset,

            goto begin_loop,
            stop)
```

Appendix C

Comoyo's Buffer Utilization

We have in this appendix included two figures demonstrating the buffer usage up to the time for the first QL degradation in the 5- and 7% packet loss scenario for Comoyo. The red line denotes the mean buffer level at the time of QL degradation. For more details see Section 5.4.2.

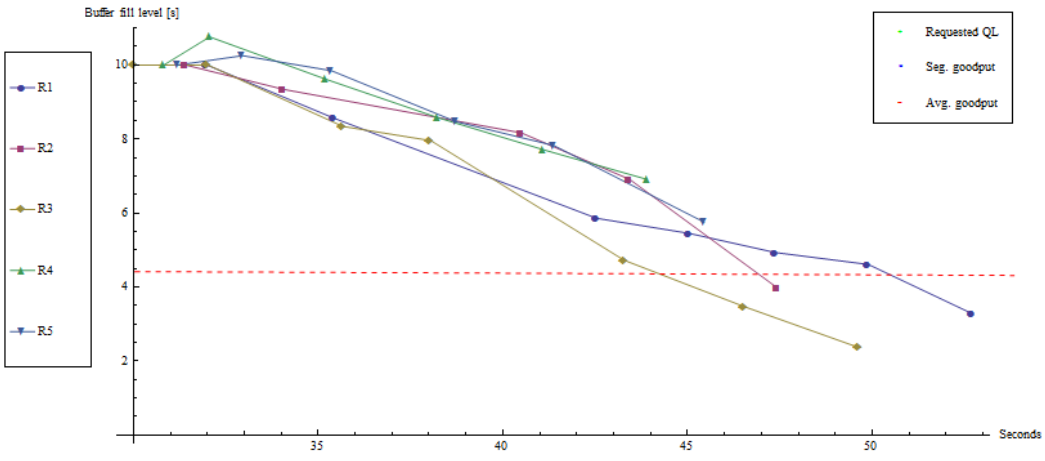

Figure C.1: Plot illustrates the buffer usage for the various runs for the 5% packet loss for Comoyo. Each point is the buffer level at the moment of a GET request. Note that the value 10s is just a reference value.

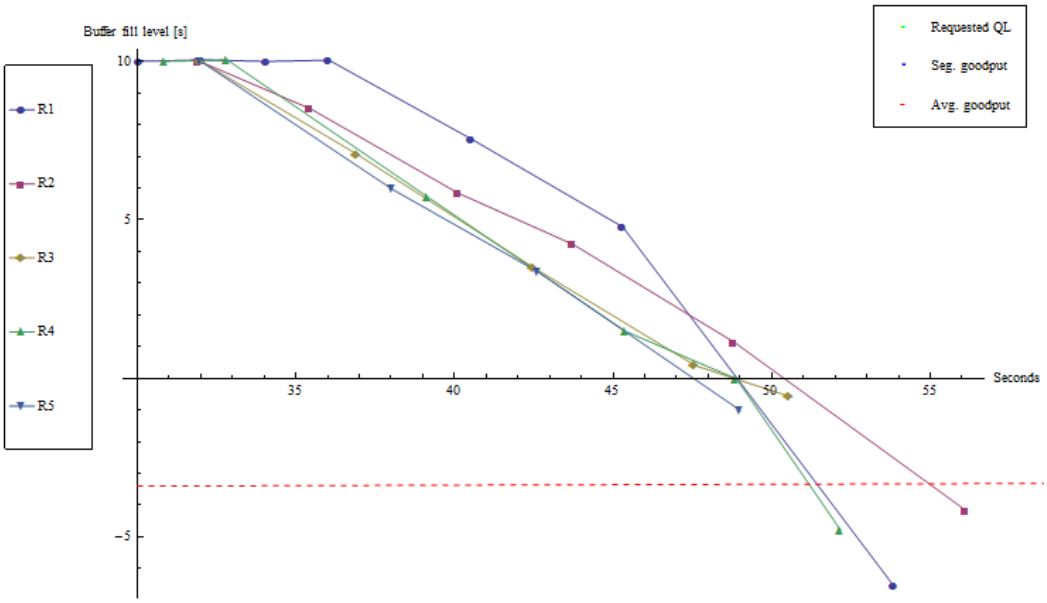

Figure C.2: Plot illustrates the buffer usage for the various runs for the 7% packet loss for Comoyo. Each point is the buffer level at the moment of a GET request. Note that the value 10s is just a reference value.

Appendix D

Mathematica

```
CreateGraphs[filePath_, qualityFileName_,
   throughputFileName_,
   accumFileName_, throughputPlotFileName_,
   throughputOverviewPlotFileName_] :=
  Module[{GETandQuality, GETandThroughput,
     GETandAccumNumPacket,
    avgTput, starttime, plots}, (
    GETandQuality =
     ReadList[filePath <> qualityFileName, {Number, Number
        }];
    GETandThroughput =
     ReadList[filePath <> throughputFileName, {Number,
        Number}];
    GETandAccumNumPacket =
     ReadList[filePath <> accumFileName, {Number, Number
        }];
    avgTput = {};
    starttime = GETandAccumNumPacket[[1, 1]];
    Do[avgTput =
      Append[avgTput, {GETandAccumNumPacket[[i, 1]],
        GETandAccumNumPacket[[i,
          2]]*1460*8/
         1000000/(GETandAccumNumPacket[[i + 1, 1]] -
          starttime)}], {i, 1, Length[
             GETandAccumNumPacket] - 1}];
    plots = GraphicsArray[
      ListPlot[GETandThroughput],
```

```
      ListLogPlot[{GETandQuality, GETandThroughput,
         avgTput},
       PlotMarkers -> {{"+", Tiny}, {"*", Tiny}, {"-",
          Tiny}},
       PlotStyle -> {Green, Blue, Red}, Ticks -> Automatic
          ,
       AxesLabel -> {"Seconds", "Mbps"}]];
    Export[filePath <> "\\" <> throughputPlotFileName,
       plots[[1]],
     ImageSize -> 72*6];
    Export[filePath <> "\\" <>
       throughputOverviewPlotFileName,
     plots[[2]], ImageSize -> 72*6];
    plots
    )];
    plots = CreateGraphs[
   "C:\\Users\\shuna\\Dropbox\\Master\\Mathematica\\The \
shit\\Sumo\\10MLimit\\No packet loss\\", "GET+QualityR2.
   txt",
   "GET+ThroughputR2.txt", "GET+accumNumPacketsR2.txt",
   "troughput10MLimitNoPLR2.png",
   "troughputOverview10MLimitNoPLR2.png"];
```

Listing D.1: Mathematica module for plotting graphs.