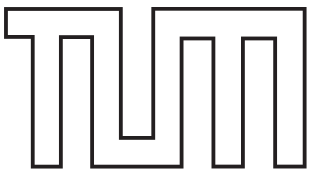


Technische Universität München
Institute for Media Technology
Prof. Dr.-Ing. Eckehard Steinbach

Master Thesis

Development and Evaluation of Dynamic Video Level Encoding Schemes for Uplink DASH in Vehicular Environments

Author:	Serhan Gül
Matriculation Number:	03637808
Address:	Traunsteiner Straße, 1 81549 München
Advisors:	Christian Lottermann Damien Schroeder
Begin:	15.04.2014
End:	15.10.2014

Abstract

The focus of the present thesis is the uplink delivery of live video using dynamic adaptive streaming over HTTP (DASH). This thesis is specifically concerned with live uplink streaming of automotive videos from the front-facing camera of a vehicle over rate-constrained heterogeneous radio access networks (RANs). DASH splits an input video in small segments each containing a few seconds of playback time and requires each segment to be available in multiple bit rates (video levels). However, the process of simultaneously creating video levels at different bit rates is computationally demanding and quickly exceeds the computational capacity of electronic control units (ECUs) of vehicles.

The major objective of this thesis is to design, implement and evaluate encoder side video level selection approaches to reduce the number of video levels that need to be encoded. Another goal is to maximize the perceptual quality of the encoded video segments by selecting the quality-optimal spatial and temporal video encoding parameters based on an objective video quality metric (VQM).

For evaluation of the developed encoder side video level selection algorithms, a reference automotive DASH scenario was created. Firstly, TCP throughput measurements were performed in vehicular mobility scenarios to characterize the network performance of the considered heterogeneous RANs. The obtained network traces were used to emulate the TCP throughput during the streaming session. Performances of three standard client adaptation algorithms in terms of user quality of experience (QoE) were investigated assuming that a static full set of 12 pre-defined video levels is available for the adaptation process. Secondly, three encoder side video level selection algorithms, which pre-filter the static full video level set according to network or client context information, were developed. To this end, two of the proposed algorithms use the measured TCP throughput information whereas the third algorithm uses the HTTP request history of the client. Performances of the proposed encoder side video level selection algorithms were compared to a reference implementation which considers the static full set of the 12 pre-defined video levels.

Results show that the proposed video level selection algorithms lead to a reduction of 67%–83% in the number of video levels that need to be encoded. At the same time, a similar QoE is achieved in terms of mean subjective quality of the delivered video segments, interrupted playback duration due to playout buffer underruns, and number of quality switches when compared to the reference implementation. These results suggest that through dynamic encoding of video levels at the server side, DASH can be successfully deployed in resource-constrained vehicular environments for uplink video delivery.

Kurzfassung

Der Fokus der vorliegenden Arbeit liegt auf der Uplink-Übertragung von Live-Videoströmen mittels Dynamic Adaptive Streaming over HTTP (DASH). Hierbei wird insbesondere die Live-Uplink Übertragung von Videoströmen aus der Frontkamera eines Fahrzeugs über ratenbeschränkte heterogene Mobilfunknetze untersucht. DASH teilt das Video in mehrere Segmente einer definierten Länge auf, die jeweils einen kurzen Abschnitt der Gesamtspielzeit enthalten und die parallele Enkodierung der jeweiligen Segmente in mehrere Bitraten (Videolevels) erfordern. Allerdings ist die parallele Erzeugung der Videolevels mit unterschiedlichen Bitraten sehr rechenintensiv und kann schnell zu einer Überschreitung der Rechenkapazität der elektronischen Steuergeräte (ECUs) von Fahrzeugen führen.

Das Hauptziel dieser Arbeit ist der Entwurf, die Implementierung und die Analyse von Videolevelauswahlalgorithmen auf der Enkodierseite, die die Anzahl der zu enkodierenden Videolevel reduzieren. Ein weiteres Ziel ist es, die subjektive Qualität der enkodierten Videosegmente durch eine optimale Auswahl der Enkodierungsparameter bezüglich der Bildqualität und der Framerate basierend auf einer objektiven Qualitätsmetrik zu maximieren.

Zur Evaluation der entwickelten Videolevelauswahlalgorithmen wurde ein Referenz-DASH-Szenario im automobilen Kontext erzeugt. Hierzu wurden zunächst TCP-Durchsatz-Messungen in Fahrzeugmobilitätsszenarien und in den berücksichtigten heterogenen Mobilfunknetze durchgeführt. Unter der Annahme, dass eine statische Menge von 12 vordefinierten Videoleveln bei jeder clientseitigen Ratenanpassung zur Verfügung steht, wurde die Performance von drei Standard-Clientadaptionsalgorithmen im Sinne von QoE untersucht. In einem weiteren Schritt wurden drei enkodierseitige Videolevelauswahlalgorithmen entwickelt, die die statische Menge von Videolevel auf Basis von Netzwerk- oder Client-Kontext-Information vorfiltern. Zu diesem Zweck verwenden zwei der entwickelten Algorithmen die gemessene TCP-Durchsatz-Information während der dritte Algorithmus die Geschichte der Client-HTTP-Anfragen verwendet. Letztlich, wurden die Performance der vorgeschlagenen enkodierseitigen Videolevelauswahlalgorithmen mit einer Referenzimplementierung verglichen, die die statische Menge von 12 vordefinierten Videoleveln berücksichtigt.

Die Ergebnisse zeigen, dass durch die vorgeschlagenen Videolevelauswahlalgorithmen eine 67%–83% Reduktion der zu enkodierenden Videolevel erreicht werden kann. Dabei wird eine ähnliche Performance wie die Referenzimplementierung erreicht, bezogen auf die durchschnittliche subjektive Qualität der gelieferten Videosegmente, die unterbrochene Playbackdauer aufgrund von Pufferunterschreitungen und die Anzahl der Qualitätsumschaltungen. Die Ergebnisse verdeutlichen, dass mit Hilfe der vorgeschlagenen Algorithmen DASH als Übertragungstechnologie für Live-Uplinkvideoübertragungen in Fahrzeugszenarien mit Rechenleistungsbeschränkten Steuergeräten realisiert werden kann.

Contents

Chapter 1

Introduction

1.1 Overview and motivation

Video streaming is today the most popular service in the Internet. According to a recent study by Cisco [Cis14], global Internet video traffic accounted for 66% of all consumer traffic in 2013. It is expected that by 2018, the share of the global Internet video traffic will have increased to 79% of the entire consumer Internet traffic.

Until roughly a decade ago, data rates and latencies of the existing mobile (cellular) networks were not deemed sufficient to stream high quality video [Ric04]. The introduction of the third generation (3G) standards such as High Speed Packet Access (HSPA) in 2000s, and more dramatically, the recent evolution towards the fourth generation (4G) standards such as Long Term Evolution (LTE) and Worldwide Interoperability for Microwave Access (WiMAX) significantly improved the throughput and low delay capabilities in mobile networks. This has led to an increasing popularity of mobile devices like smartphones and tablet PCs for streaming applications such as video streaming. It is expected that the global mobile data traffic will increase 11-fold between 2013 and 2018 and a large part of this traffic will be in the form of video [Cis14].

In addition to the capacity improvements in mobile networks, enhanced hardware and signal processing capabilities of mobile devices equipped with cameras have enabled upstreaming of user-generated rich multimedia content to video portals in real time [Ess14]. An important use case of live uplink video streaming is the domain of vehicular applications which target at improving road safety, traffic control and public security [VBBH14]. For example, the front-facing camera of a vehicle can be used in real-time street view [GOMF12, LGSS] or remote traffic monitoring systems [VBBH14]. Further vehicular video streaming applications include in-vehicle video surveillance, overtaking assistance and public transportation assistance [BVJS14]. Figure 1.1 shows an overview of some of the prospective vehicular video streaming applications.

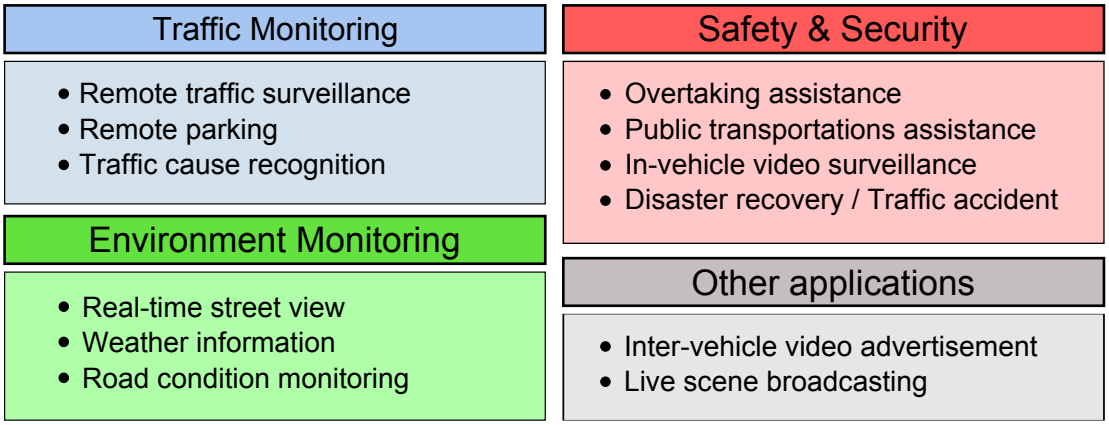


Figure 1.1: Prospective applications of video streaming in vehicular environments.

Despite the improved capabilities of mobile networks, bandwidth is still an expensive and scarce resource in wireless communications. It is well known that there is a theoretical upper limit on the rate of information that can reliably be transferred over a communications channel [Sha48]. Also, the available frequency spectrum is finite and shared by multiple wireless networks and users. Furthermore, automotive applications always have to deal with unpredictable network performance due to fluctuations of the available throughput and usage of diverse heterogeneous radio access networks (RANs) [MTKM09]. For these reasons, resource-hungry video streaming applications have to utilize the available bandwidth as efficiently as possible [MLT12], and in-session rate adaptation might be required [CMP11].

Recently, there has been a paradigm shift towards *adaptive* bit rate video streaming as it has several benefits compared to classical, non-adaptive video streaming. Also, due to various deployment advantages, Hypertext Transfer Protocol (HTTP) over Transmission Control Protocol (TCP) has gained popularity as the standard delivery protocol[1] [Sto11]. Consequently, an increasing number of video applications have started to employ adaptive HTTP streaming (AHS) instead of non-adaptive streaming. Figure 1.2 compares non-adaptive vs adaptive HTTP streaming and illustrates their respective video delivery mechanisms.

In 2012, the Moving Pictures Expert Group (MPEG) published the first international AHS standard under the name DASH, also known as MPEG-DASH [3GP11, ISO12]. DASH adapts the quality of the delivered video to users' resources by offering multiple encoded versions of the same video (e.g., in different bit rates or resolutions) which are temporally split into small segments of a defined length, usually in the order of seconds. The different bit rate encodings of a video segment are called *video levels*. In contrast to non-adaptive streaming, the current video bit rate can be adapted dynamically to changing network and server conditions. The streaming session is fully controlled by the client, which monitors

[1] See Section 2.4.3 for an overview of the reasons for the popularity of HTTP.

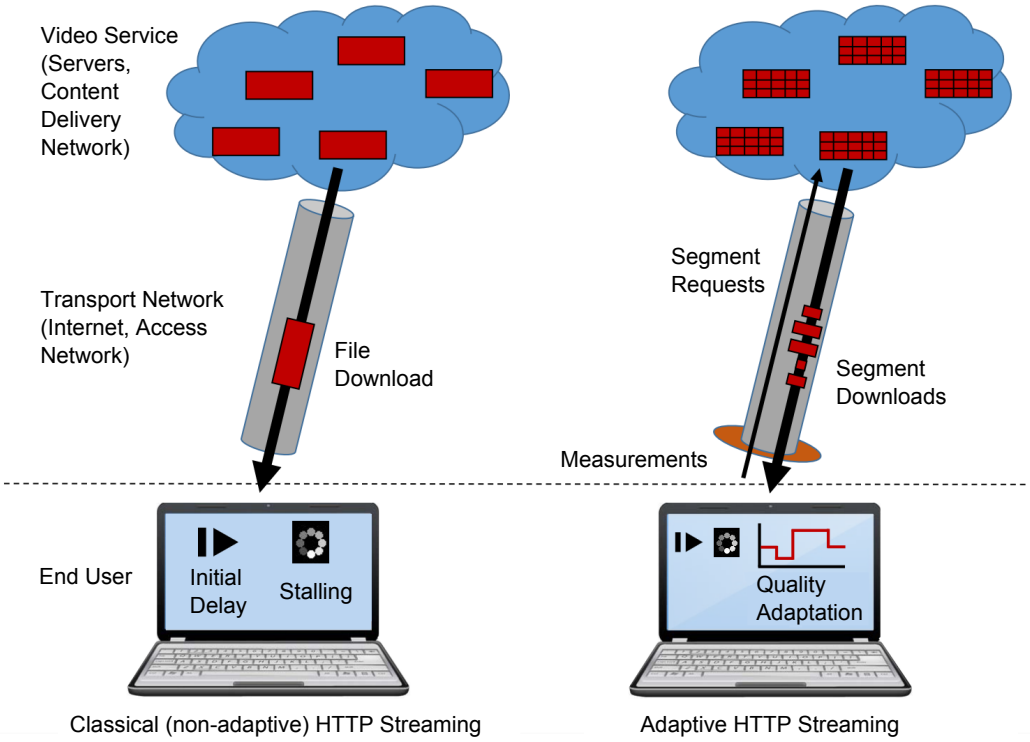


Figure 1.2: Comparison of non-adaptive HTTP streaming and adaptive HTTP streaming (AHS) [SES+14].

its network conditions continuously [Sto11]. Based on network measurements and other client-specific parameters (such as client buffer fullness level), the adaptation algorithm at the client side requests the successive media segments dynamically at a bit rate that matches the available network performance. Thereby, playout interruptions are reduced and the subjectively perceived video quality is significantly improved [SES+14].

Despite its adaptivity to changing networking conditions, its efficient bandwidth usage, and the improved perceptual video quality, DASH (or in general AHS) has not yet been considered for uplink video delivery from vehicles [LGSS]. This is mainly due to a number of challenges:

Firstly, an accurate, *low-complexity* video quality model is required to estimate the subjective quality of the video stream and determine the best possible trade-off between spatial and temporal encoding parameters in order to maximize the perceptual quality for the viewer [LMS+14]. Extraction of content-dependent video parameters from source video is computationally complex and requires processing power which might not be available in a vehicle. As a remedy, Lottermann et al. proposed a low complexity approach for the estimation of content-dependent video parameters based on the *context* information of a vehicle [LMS+14], i.e., the environmental information captured by the vehicle's sensors. Using their approach, the perceptual quality can be estimated and modeled at a low

complexity for videos captured with a front-facing camera of a vehicle, without having to access the raw video.

Secondly, video encoding parameters need to be adjusted for each target bit rate in order to meet the target bit rate on the encoder side. This is typically achieved by rate control algorithms. However, most state-of-the-art rate control algorithms either ignore the effect of frame rate changes [LLS07] or require direct access to the source video and calculation of computationally-complex, content-dependent parameters [WMO09, MXOW12]. As a remedy, Lottermann et al. proposed a low complexity bit rate model [LSS14], which is based on the quantization parameter (QP), frame rate as well as temporal and spatial activity measures that can easily be calculated from uncompressed source video, or estimated using camera context information [LMS+14]. Their model is suitable for deployment in live vehicular video streaming applications.

Another crucial challenge for vehicular live uplink DASH is the creation of different video levels at the encoder side. Due to limited computational resources, the DASH encoder deployed in a vehicle might not be able to create the same number of video levels in parallel as content delivery networks (CDNs) [LGSS]. To mitigate this, the present thesis proposes network- and client-aware encoder-side video level selection approaches which reduce the number of video levels to be encoded at the source side. The goal of the proposed approaches is to dynamically filter a full, static set of video levels at the encoder-side and create a reduced set of video levels, which is then presented to the client for client-side rate adaptation.

To this end, an automotive uplink streaming scenario is created through TCP uplink throughput measurements in real HSPA and LTE networks. First a reference scenario is considered, in which the full, static set of video levels are used, and no encoder-side adaptation is performed. Then, three encoder-side video level selection algorithms are developed. The first two algorithms are based on the network-context information received from the heterogeneous RAN modem of the vehicle whereas the third algorithm employs the video level request history of the DASH client. These encoder-side algorithms are implemented on top of standard AHS client adaptation algorithms. A simulation framework is developed to investigate the performances of the proposed algorithms in terms of user quality of experience (QoE).

The results show that the proposed video level selection approaches reduce the number of video levels that need to be encoded by 67%-83%, and thus the encoding complexity significantly while achieving a similar QoE as a reference scenario which uses a full video level set. The main advantage of the approaches proposed in this thesis is that the mechanisms and deployment characteristics of DASH-based HTTP/TCP streaming can be used while achieving low computational demands for the DASH encoder. It is expected that the proposed methodology can also be applied to upstream live video from other devices with limited computational capacity, such as smartphones and surveillance cameras.

1.2 Contributions

The present thesis has two main contributions:

1. A reference automotive streaming scenario is constructed for uplink delivery of vehicle camera streams captured with a front-facing camera. Specifically:
   - Vehicular on-board and off-board components and architecture are defined.
   - Network performance of heterogeneous RANs in automotive scenarios is characterized via uplink and downlink TCP throughput measurements.
2. An automotive uplink DASH architecture is proposed. Specifically:
   - Three state-of-the-art AHS client rate adaptation algorithms are implemented, and their performance is evaluated in terms of user QoE.
   - Three encoder side video level selection algorithms are developed for live uplink video streaming in vehicular environments. Performance of the algorithms is evaluated in terms of user QoE and compared to a reference implementation which uses a static, full set of video levels.

1.3 Organization

The rest of this thesis is structured as follows. Chapter 2 provides background material for this thesis by reviewing the relevant topics of video processing and communication. Chapter 3 provides a literature review of the state-of-the-art AHS systems, client side rate adaptation algorithms, objective video quality metrics (VQMs) as well as the on-board and off-board vehicular streaming architectures. Chapter 4 presents the automotive TCP throughput measurements performed in real HSPA and LTE networks. In Chapter 5, a vehicular DASH architecture is proposed to upstream videos from a vehicle's front-facing camera. Three state-of-the-art AHS client rate adaptation algorithms are implemented, and their performance in terms of user QoE is investigated based on the network traces from Chapter 4. In Chapter 6, three encoder side video level selection algorithms are developed, and their performances in a vehicular DASH architecture are investigated. Finally, Chapter 7 concludes this thesis and points out some limitations and future research directions.

Chapter 2

Theoretical background

This chapter reviews basic concepts of video compression, quality of experience (QoE), and mobile multimedia streaming that form the prerequisite material for this thesis.

2.1 Video compression basics

Video compression is an important application of digital video processing. The goal of video compression is to reduce the data rate of a video sequence such that the transmission of a video over a given communication channel is feasible [WOZ02]. Compression will continue to be a key technology for digital transmission and storage of video as long as the availability and demand for video keep outpacing the available network capacity [GGKV98].

Video compression takes advantage of the inherent statistical redundancy in video signals and exploits limitations of the human visual system (HVS) to omit signal components which are perceptually not significant [GGKV98]. Compression can be *lossless* which allows the perfect reconstruction of data without any loss of information. Two prominent lossless compression (entropy coding) techniques are Huffman coding [H$^+$52] and arithmetic coding [WNC87]. However, compression performance of lossless techniques are limited. Hence, *lossy* compression techniques are used to achieve higher compression rates at the cost of imperfect source reconstruction. In this case, there is a trade-off between the number of bits in the representation (rate) and the fidelity of the representation (distortion) [OR98].

2.1.1 Rate-distortion models

The starting point of classical rate-distortion theory is Shannon's seminal work [Sha48]. The goal of rate-distortion theory is to represent a source with fewest number of bits possible for a given reproduction quality [OR98]. This reproduction quality is denoted as

distortion and is commonly measured by peak signal-to-noise ratio (PSNR) (2.2), which is the ratio between the maximum possible power of a signal and the power of corrupting noise. PSNR depends on the mean squared error (MSE) (2.1) between an original and an impaired image or video frame [Ric04]. In image and video processing applications, MSE is calculated by performing a pixel-to-pixel comparison of the original image X and a reconstructed image $\hat{X}$ [Ess14].

$$\text{MSE} = \frac{1}{M \cdot N} \sum_{m=1}^{M} \sum_{n=1}^{N} (X(m,n) - \hat{X}(m,n))^2 \tag{2.1}$$

where M and N are the height and width of the image, respectively. Hence, PSNR is defined as:

$$\text{PSNR} = 10 \log_{10} \frac{(2^B - 1)^2}{\text{MSE}} \,\text{dB} \tag{2.2}$$

where B is the number of bits per image (video frame).

Due to its simplicity, PSNR is widely used to evaluate the distortion of decompressed video frames. However, PSNR is criticized for being poorly correlated with the characteristics of HVS [Gir93]. A particular PSNR value does not necessarily reveal the true subjective quality. As an example, Figure 2.1 shows an uncompressed image and three distorted versions of it. In Figures 2.1b and 2.1c, blurring is applied overall and images with PSNR values of 30.6 dB and 28.3 dB (relative to the original) are obtained. On the other hand, in Figure 2.1d, only the background of the image is blurred and the resulting image has a PSNR of 27.7 dB, which is lower than both Figures 2.1b and 2.1c. However, most viewers would rate Figure 2.1d significantly better than Figure 2.1c since the face is clearer. This example shows that true subjective quality might contradict PSNR ratings in some cases [Ric04].

Another limitation of PSNR is its content dependency. The authors of [HTG08] showed that PSNR is an unreliable VQM when quality across various video contents is compared. Nevertheless, PSNR is a good measure of distortion when content and codec are fixed while comparing two videos [HTG08].

Structural Similarity Index (SSIM) offers a higher match with the perceived quality compared to PSNR by considering structural information, i.e., inter-dependencies between pixels especially when they are spatially close [WBSS04]. SSIM compares the luminance, contrast, and structure of two images after mean subtraction and variance normalization [Ess14]. However, both PSNR and SSIM only consider spatial video quality impairments. Video coding techniques[1] such as scalable video coding (SVC) and multiple bitrate coding (MBR) vary the video quality not only with QP, but also with the frame rate. PSNR and SSIM do not accurately reflect the perceptual quality when changes in frame rate are involved. Moreover, the correlation between PSNR and subjective quality becomes lower

[1] See Section 2.3 for an overview of video source rate adaptation techniques.

(a) original (b) 30.6 dB (c) 28.3 dB

(d) Image with blurred background (27.7 dB)

Figure 2.1: Comparison of the original and distorted versions of an example image with different PSNR values [Ric04].

when frame rate and QP are jointly adjusted [FSWV07]. Hence, PSNR and SSIM are not optimal quality metrics for video sequences coded with variable QPs and frame rates.

Additionally considering temporal impairments is another recently proposed technique in video encoding. For example, Wang et al. proposed a VQM which combines the spatial SSIM metric with an additional motion analysis stage [WLB04]. Pinson and Wolf developed another VQM which extracts *quality features* that capture spatial, temporal, and color-based differences between the original and distorted videos [PW04]. Other objective VQMs, which consider both spatial and temporal impairments, have been studied in the literature [FSWV07, OMW09, PS11]. They will be briefly discussed in Section 3.3.2.

2.1.2 Content-dependent video parameters

In order to quantify the spatial and temporal information of a video sequence of a certain length, two parameters are recommended by the International Telecommunication Union (ITU) [IR07]. Slightly modified versions of these parameters are defined as spatial activity (SA) and temporal activity (TA) in [PS11] and given by:

$$\mathrm{SA} = mean_{\mathrm{time}}\{std_{\mathrm{space}}[Sobel(F_n)]\} \tag{2.3}$$

$$\mathrm{TA} = mean_{\mathrm{time}}\{std_{\mathrm{space}}[F_n - F_{n-1}]\} \tag{2.4}$$

where F_n is the video frame (luminance plane) at time n and *Sobel* filter [SF68] is a discrete differentiation operator used in edge detection applications. SA is a measure that indicates the amount of spatial detail in a video. To compute the SA value, each video frame is first filtered using a Sobel edge detector. For each Sobel-filtered frame in the video sequence, standard deviation over all pixels is computed. This results in a time-series of spatial information values which are then averaged to obtain the SA value of the video.

TA is a measure that indicates the amount of temporal changes of a video sequence. It tends to be higher for high motion sequences. To compute the TA value, the difference between the pixels at the same position belonging to two subsequent frames is computed for the whole video sequence. Then, TA is computed as the mean over time of the standard deviation of this difference over all pixels [IR07].

2.1.3 Group of Pictures (GoP)

In video coding standards, frames are partitioned into sequences called groups of pictures (GoPs). A GoP is an encoding of a sequence of frames that can be decoded entirely within the same GoP. Frames in a GoP are classified according to whether temporal prediction is applied during their encoding [WOZ02].

If a frame is encoded independently of all the other frames, it is called an *intra-frame* or I-frame. The first frame of a GoP is always coded as an I-frame. A *predictive-coded frame* or P-frame contains only the differences relative to previously decoded frames. In MPEG-1 [ISO93] and MPEG-2 [ISO00] standards, a P-frame can only reference one other frame (an I-frame or another P-frame) and that frame must precede the P-frame in display and decoding order. Finally, a *bi-predictive coded frame* or B-frame uses bi-directional prediction, i.e., encodes the differences between the current frame and both the preceding and following frames. In MPEG-1 and MPEG-2, a B-frame can only reference two other frames, one of which must precede and the other must follow the B-frame in display order. Also, all referenced frames must be I- or P-frames. In newer standards such as H.264/MPEG-4 AVC, multiple previously decoded frames (up to 16) can be used as references for decoding P- and B- frames. Moreover, B-frames can also be used as references while coding other P- or B-frames [WOZ02].

A typical GoP conforming to MPEG-1 and MPEG-2 standards is displayed in Figure 2.2. It starts with an I-frame, followed by interleaving P- and B-frames. I-frames typically require more bits to encode than P- or B-frames, and they create random access points, i.e., one can access any GoP without decoding previous GoPs. A GoP is characterized by the number of concurrent B-frames m for a given GoP length n [LS14]. Accordingly, the GoP in Figure 2.2 has $m = 2$ and $n = 9$.

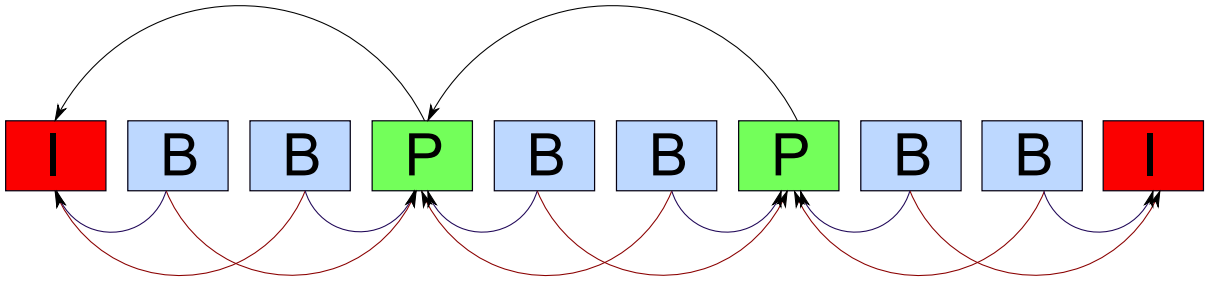


Figure 2.2: Group-of-Pictures (GoP) structure in MPEG-1 and MPEG-2 standards.

2.2 Quality of Experience (QoE)

QoE is a multidisciplinary measure based on cognitive science, social psychology and engineering, focused on understanding overall human quality requirements [LC12]. There are different definitions of QoE across current ITU, European Telecommunications Standards Institute (ETSI) and other telecommunications literature. ITU defines QoE as "the overall acceptability of an application or service, as perceived subjectively by the end-user" [ITU08b]. On the other hand, ETSI defines QoE as "a measure of user performance based on both objective and subjective psychological measures of using an ICT service or product" [ETS10]. Different from ITU, ETSI's definition takes also the objective psychological measures into account which do not rely on the opinion of the user. For example, these measures might be context variables like task completion time measured in seconds or task accuracy measured in number of errors.

Another well-established quality measure with a much longer history than QoE is quality of service (QoS). ITU defines QoS as "totality of characteristics of a telecommunication service that bear on its ability to satisfy stated and implied needs of the user of the service" [ITU08a]. Although the ITU definition refers to user satisfaction, the term QoS is mainly used to define the technical parameters of the telecommunications systems (e.g., network delay, packet loss and jitter) in the engineering literature [ETS10]. Therefore,

QoE has emerged as an important concept to measure user satisfaction. In this sense, QoE is a set of human-centric factors whereas QoS emphasizes the technology-related parameters [LC12]. However, there is a strong dependency between the two concepts. The effect of the network QoS parameters on QoE can be modelled through analytical and empirical methods [MCC11]. Hence, in a video streaming system, it is important to monitor the network transport QoS parameters which ultimately contribute to the user's QoE [CSRK11]. Furthermore, QoE level of a service needs to be monitored in order to detect potential bottlenecks in an operator's communication network. With this knowledge, one or more QoS parameters can be tuned to achieve a higher user satisfaction [ETS10].

2.3 Video coding techniques

Video streams are either scalable or non-scalable. Scalability refers to the capability of recovering *meaningful* video content by decoding only partial compressed bit streams [WOZ02]. State-of-the-art video coding standards used to encode scalable video streams are the scalable extensions of H.264/AVC [ISO03] and H.265/HEVC [ISO13]. Different techniques exist for adapting the video source bit rate of scalable and non-scalable streams, respectively. These techniques can be classified into three main categories: transcoding, scalable video coding (SVC), and multiple bitrate coding (MBR), also known as stream switching [CMP11]. These video source rate adaptation techniques are illustrated in Figure 2.3.

Transcoding For non-scalable video, a separate transcoder can be used to realize a certain target bit rate. Transcoding offers a very fine-granularity by adapting the video encoding parameters (e.g., frame rate, QP, and resolution [MFW13]) to match the available network capacity. However, transcoding introduces additional complexity and encoding delays since the input stream has to be decoded and re-encoded at a target rate [Ess14]. Additionally, it has poor scalability[2] and is difficult to deploy in CDNs since it has to be done on a per-client basis [CMP11].

Scalable video coding The design goal in scalable coding is to minimize the reduction in coding efficiency while creating scalable bit streams. SVC creates inherently scalable video streams by encoding the video into a base layer (BL) and several enhancement layers (ELs) [SMW07]. The base layer can be independently decoded at the receiver and provides a minimum video quality. Enhancement layers can be added to improve the perceived quality. Scalability can be employed in temporal, spatial or quality domains, i.e., the base layer can be encoded at a lower frame rate, a lower spatial resolution or a lower fidelity (PSNR). A combination of these scalability modalities is also possible [WOZ02]. Compared

[2]Here scalability refers to the capability of a system to handle an increasing amount of load.

to transcoding, SVC reduces processing costs since the raw video is encoded only once and adapted on-the-fly by exploiting the layered structure. However, SVC brings additional coding complexity and requires specialized servers to run its rate adaptation logic which depends on the employed codec. Also, scalable video content cannot be cached in standard proxies [CMP11].

Multiple bit rate coding Another option is to encode the same content at multiple bit rates to provide redundant representations. In most implementations, the video is partitioned into segments of a defined length, all of which have many versions encoded at different bit rates [Ess14]. Thus, rate adaptation is performed at the client side by switching between pre-encoded segments, and no further processing costs are required. Another advantage of MBR is that it does not rely on the particularities of the employed codec, i.e., it is *codec-agnostic* [CMP11]. A recent and prominent example of MBR is adaptive HTTP streaming (AHS) which is explained in Section 2.4.3.

2.4 Mobile multimedia streaming

Mobile multimedia streaming refers to the transmission of live or pre-encoded media content from a content generator (server) to one or multiple consumers (clients) over a wireless network [Ess14]. In terms of the number of clients, streaming can be classified as point-to-point, multicast, and broadcast [ATW02]. In point-to-point video streaming, there is one server and one client (unicast video streaming). Video conferencing is an example of point-to-point streaming. Broadcast video streaming involves one server and multiple clients (e.g. satellite digital television broadcast). Multicast video streaming is similar to broadcast video streaming but with less number of clients. An example of multicast streaming is the IP-Multicast video streaming over the Internet [WKLC09].

Two application layer protocols are widely used for mobile multimedia streaming: Hypertext Transfer Protocol (HTTP) [FGM+99] and Real-time Transport Protocol (RTP) [SCFJ03]. They are typically used over the transport layer protocols Transmission Control Protocol (TCP) and User Datagram Protocol (UDP), respectively. In the following, RTP/UDP and HTTP/TCP-based mobile multimedia streaming approaches are reviewed.

2.4.1 RTP-based streaming

RTP is a streaming protocol that provides end-to-end network transport functions suitable for multimedia applications over unicast or multicast network services. RTP commonly runs on UDP, a transport protocol without any inherent rate-control mechanisms [BAB11]. RTP is *push-based*, i.e., the server uses RTP/UDP protocol to push the media stream to the client once the connection is established. Streaming of the packets to the client continues

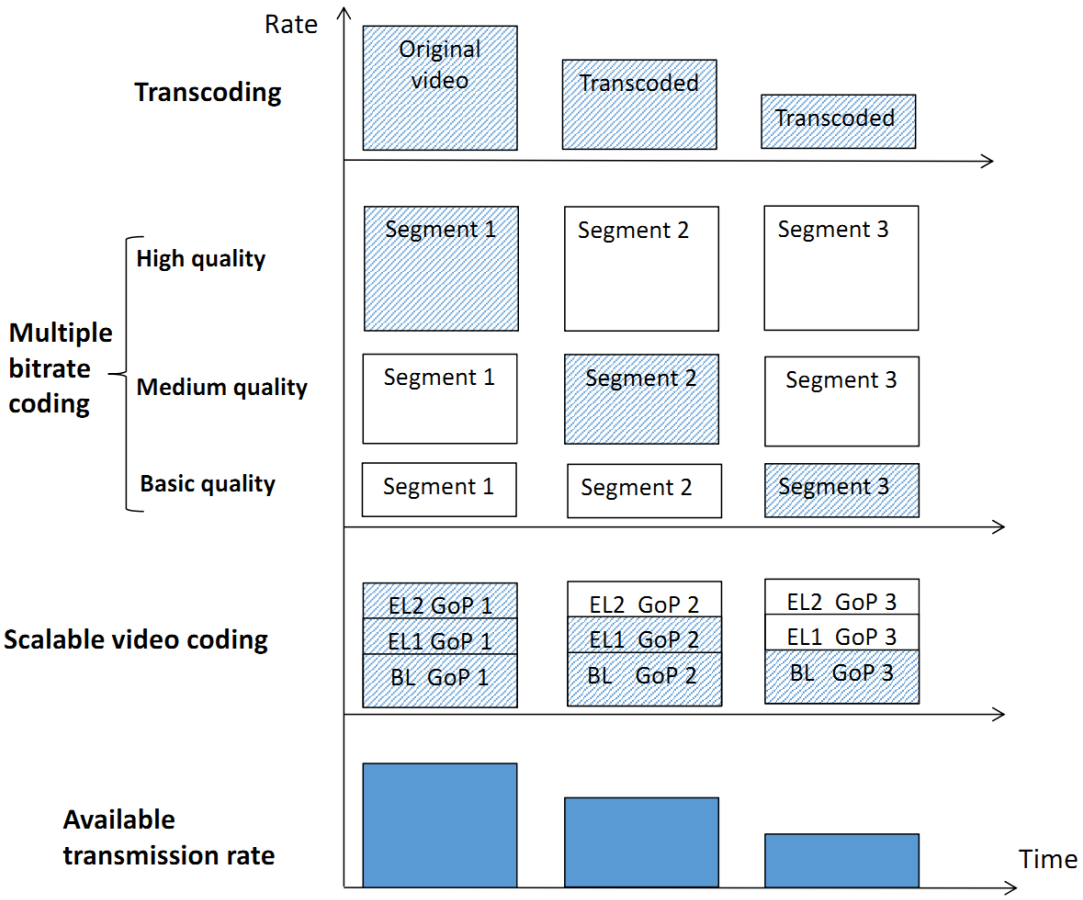


Figure 2.3: Illustration of different video source rate adaptation techniques. Transcoding changes the video bit rate by adapting the video encoding parameters. Multiple bit rate coding (MBR) encodes the same video content at different video quality levels. Switching decisions are made after every segment. Scalable video coding (SVC) encodes each GoP into a BL and multiple ELs. Shaded regions show the transmitted videos [Ess14].

until the client stops or interrupts the session. A session control protocol called Real-time Streaming Protocol (RTSP) [SRL98] is used to maintain the session on the application layer.

In RTP/UDP streaming, the server pushes packets to the client at a bit rate adapted to the transmission characteristics between the client and the server. The client usually performs bandwidth monitoring to compute network metrics such as round trip time (RTT), packet loss, and network jitter periodically. It communicates this information via receiver reports over the RTP Control Protocol (RTCP) to the server which makes the corresponding rate adaptation decisions [BAB11].

RTP/UDP streaming uses the bandwidth efficiently since it uses a purpose-built protocol

(RTSP) and built-in algorithms designed for tasks such as bit rate management, retransmission, and content caching. Moreover, it supports multicast delivery which lets the server send a packet only once to a group of clients requesting that packet [BAB11]. However, RTP/UDP streaming has several disadvantages [Sto11]. First, packet losses due to the inherent unreliability of UDP might cause severe video quality degradations. Second, it requires a specialized server that implements RTSP. Also, firewalls often filter UDP packets. Lastly, flow control, congestion control, and error recovery duties are left to the application layer since UDP does not implement these protocol mechanisms. The computational demand increases with the number of streaming clients since a separate encoding entity is required for each client [Ess14].

2.4.2 Progressive download over HTTP

HTTP is an application layer protocol which uses the request-response method in the client-server computing model. Progressive download over HTTP is a widely used media delivery method in which the client sends HTTP request to the server and starts receiving content from it as quickly as possible [ABD11], i.e., it is a *pull-based* streaming approach different from RTP/UDP streaming. The client downloads parts of a video file before the beginning of the actual playout. As soon as the client buffer reaches a minimum required level, video playout starts. In contrast to the classical file download, the client does not have to wait until the download of the video file is completed to start the playout.

Since HTTP uses TCP as the underlying transport protocol, packet losses are handled by TCP's retransmission mechanisms ensuring reliable delivery. Also, usage of TCP (specifically HTTP over TCP) greatly simplifies the traversal of firewalls and network address translations (NATs) [ABD11]. Moreover, TCP guarantees the stability of network by means of an efficient congestion control algorithm [CMP11]. However, progressive download suffers from the late delivery of packets and TCP rate fluctuations which might cause playout interruptions. Client side buffering and an initial playout delay are used to compensate for those issues [Ess14].

Despite its popularity, progressive download has many shortcomings [Sto11, ABD11]. One of the major disadvantages of progressive download is that all clients receive the same encoded video, and a client cannot change its download rate without stopping and restarting the download, i.e., it is *non-adaptive*. This is a major issue since the available bandwidth might vary significantly over the course of the download both across different clients and across time for the same client [ABD11]. Therefore, playout freezes can occur if the bandwidth is limited. Besides that, the bandwidth might be wasted if the user decides to stop watching the content or switch to another content. Moreover, progressive download does not support live streaming [Sto11].

2.4.3 Adaptive HTTP streaming (AHS)

Until recently, the main consensus among the video streaming community was that TCP could never serve as a feasible transport protocol for video streaming due to the throughput variations caused by its congestion control and potentially large transmission delays. Consequently, UDP has been adopted as the underlying transport protocol for a long time in earlier video streaming research and applications [ABD11]. However, in the last few years, it has been shown that TCP does not necessarily exacerbate the performance of video streaming applications, especially if the video player is able to adapt to large throughput variations. A TCP-based streaming protocol, progressive download over HTTP, was used by the first generations of HTTP/TCP-based video streaming applications. However, it has many shortcomings as explained in Section 2.4.2. Another pull-based streaming protocol

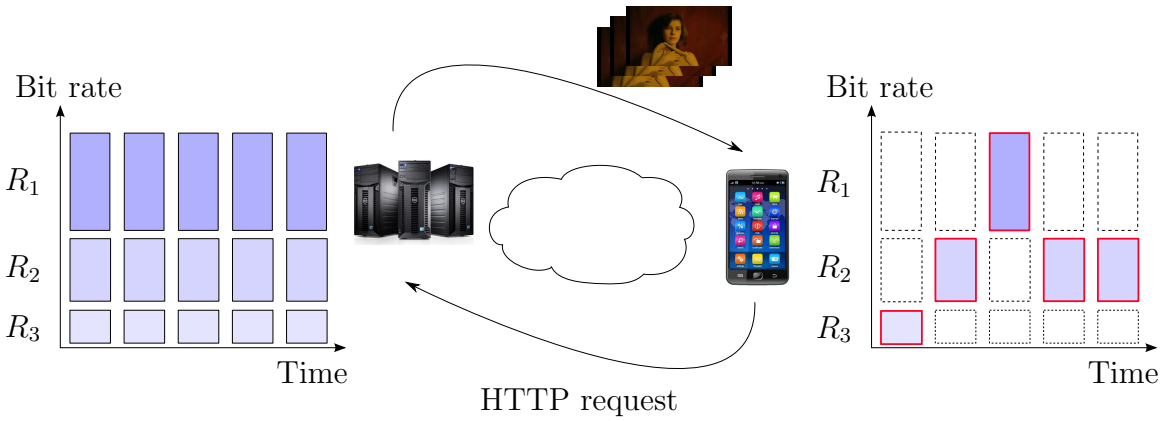


Figure 2.4: Overview of a typical AHS system. The server stores pre-encoded segments at different bit rates. Client requests the appropriate segment depending on the available throughput and its buffer status. Adaptation decisions are made at segment boundaries.

that runs over TCP is adaptive HTTP streaming (AHS). Instead of performing one large file download, AHS splits the video stream into segments of a defined length (typically 2 to 10 seconds [EKR+14]) which are downloaded individually. The server encodes segments at multiple bit rates representing different video quality levels. Segment information such as timing, video resolution and bit rates is presented to client in a manifest file. A client can continuously monitor its available throughput and buffer status, and request the media segment that best matches them [Ess14, Rii13]. When the client plays the requested segments consecutively, the original media stream can seamlessly be reconstructed [BAB11]. Figure 2.4 illustrates a typical AHS scenario.

AHS has many advantages over RTP/UDP streaming due to the choice of HTTP/TCP as the delivery protocol [Sto11]. First, it can re-use the existing standard HTTP-servers and caches. Thus, an ordinary web server can stream the media content, and no specialized server is required as in RTP/UDP streaming. Secondly, HTTP-based streaming does not suffer from firewall filters since almost all network nodes in the Internet are configured to

support HTTP traffic. Thirdly, the client has full control of the streaming session meaning that it can seamlessly change video bit rate on-the-fly in reaction to changes of the available bandwidth without requiring any negotiation with the streaming server.

Despite its many benefits, AHS brings also some disadvantages compared to the push-based streaming protocols such as RTP/UDP streaming. AHS is usually less efficient than RTP/UDP streaming since RTP imposes a lower transmission overhead than HTTP/TCP streaming [BAB11]. Furthermore, the complex interaction between TCP's congestion control and the application-layer rate adaptation mechanisms creates a nested double feedback loop. Dynamics of such interacting control systems are hard to predict [ABD11]. Lastly, as a pull-based streaming protocol AHS provides only unicast delivery and does not have multicast support as in push-based streaming protocols [BAB11].

Chapter 3

Related work

This chapters reviews the state-of-the-art AHS systems, client rate adaptation algorithms and objective video quality metrics used to assess the perceptual video quality. Also, on-board and off-board vehicular streaming architectures are presented.

3.1 AHS systems

Recently, HTTP has become the de facto protocol for adaptive streaming of the video content [EKR+14]. AHS is now widely deployed in many proprietary implementations for example by Microsoft [Zam09], Adobe [Ado10], and Apple [PM10]. More recently, YouTube started to support AHS by offering *Auto* mode in their video quality selection [TL13]. Moreover, AHS has been standardized by MPEG under the name DASH, also known as MPEG-DASH, in order to provide interoperability between devices and servers of various vendors [ISO12]. In the following, the main components of the MPEG-DASH standard are explained, and the rate adaptation mechanisms of the existing proprietary AHS implementations are briefly discussed.

MPEG-DASH

In AHS, the same media content encoded with various bit rates must be available at the server. This media content might consist of different components such as audio, video, and text. MPEG-DASH describes these characteristics of the media content in an index file called Media Presentation Description (MPD) which is an extensible markup language (XML) file.

Figure 3.1 displays the hierarchical MPD structure. MPD consists of multiple presentation *periods* with certain start and end times. Each period contains one or more adaptation

sets. An adaptation set provides information about different media components, e.g. audio or video. For example, one adaptation set might contain different bit rates of the video content whereas another one might contain different bit rates of the audio content. Many representations of a certain kind of media content exist in each adaptation set. Assuming that the media content is video, an adaptation set contains multiple representations of the video encoded at different bit rates or resolutions. Bit rates of the different representations are called *video levels*. A representation contains multiple segments in time domain, each of which has a separate universal resource locator (URL) and can be downloaded using HTTP GET requests [Sod11].

At the beginning of the streaming session, a DASH client receives the MPD. Then, it parses the MPD, builds a timeline, and plays the video by requesting the appropriate segments at each segment boundary. The DASH client implements a rate adaptation algorithm which selects the appropriate segments based on information such as the available TCP throughput and the client buffer status. Those adaptation algorithms are not a part of the DASH standard; thus they are vendor specific [Sto11]. Three state-of-the-art rate adaptation algorithms are described in Section 3.2.

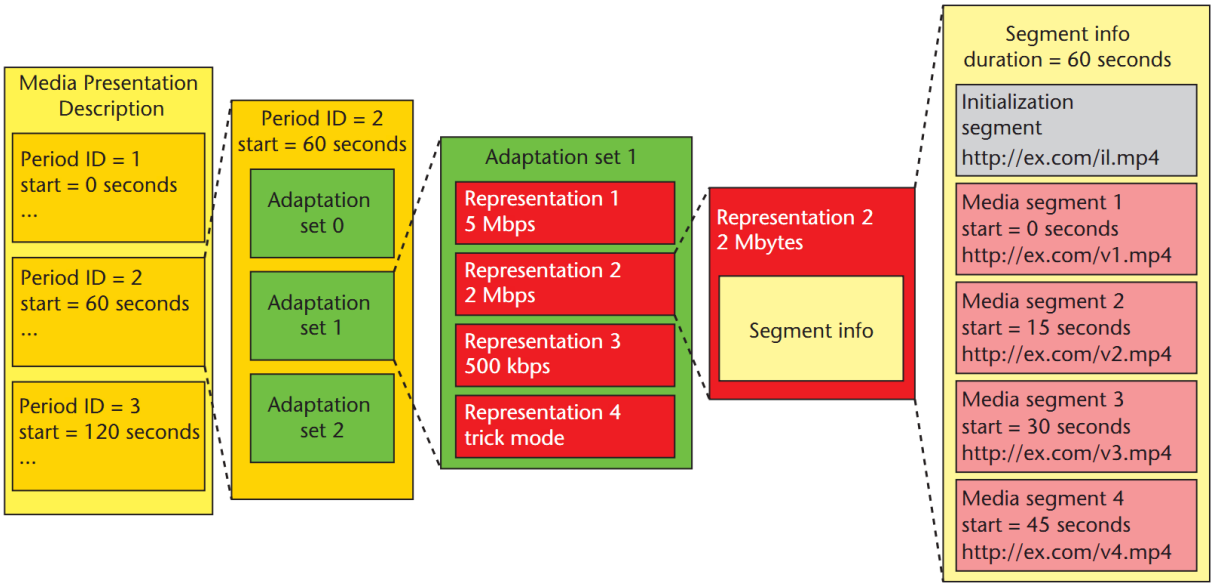


Figure 3.1: Media Presentation Description (MPD) for dynamic adaptive streaming over HTTP (DASH) [Sod11].

Proprietary AHS implementations

Akhsabi et al. [ABD11], Müller et al. [MLT12], Riiser et al. [RBV+12], and Evensen et al. [EKR+14] investigated the performances of Microsoft Smooth Streaming (MSS),

Apple Live Streaming (ALS), and Adobe HTTP Dynamic Streaming (HDS) over wireless networks focusing on how they react to persistent and short-term available throughput fluctuations.

Evensen et al. used six different quality levels (ranging from 250 to 3000 kbit/s) and compared the performances of MSS, ALS, and HDS over commercial 3G networks. They concluded that these proprietary systems follow very different rate adaptation strategies and they all have shortcomings like playout interruptions and frequent quality switches [EKR+14]. Figure 3.2 displays the obtained video quality level index as a function of time over a bus route in Oslo along with the observed bandwidth.

Evensen et al. observed that Adobe's player always selects the quality level that most closely matches the current bandwidth. This leads to a lot of video quality fluctuations and playout interruptions. On the other hand, Apple's player aims to avoid playout interruptions at all costs. It also provides relatively high bandwidth utilization compared to HDS and MSS. However, its buffer size is very large (approximately 200-250 seconds [MLT12]). Evensen et al. rates Microsoft's player as the best performer among the commercial media players. MSS has fairly good average quality and does not switch very frequently between different video levels. It also acts very conservatively and uses a stepwise switch-up approach like ALS, but its step size is smaller than that of ALS [MLT12]. However, MSS also requires a lot of improvement in terms of bandwidth utilization and playout interruptions.

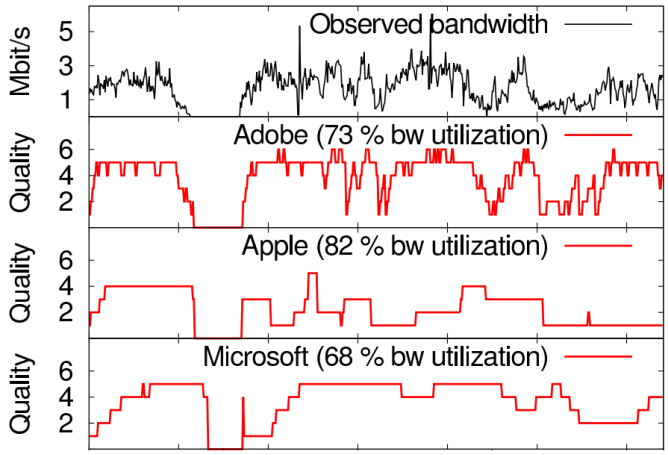


Figure 3.2: A performance comparison of the proprietary AHS implementation in a vehicular mobility scenario [EKR+14].

AHS in vehicular mobility scenarios

AHS is required in environments in which the network performance is time-varying and unpredictable. This holds true especially in automotive environments since the vehicle moves at different speeds and connects to various heterogeneous RANs. In the literature, several studies exist which investigate the influence of vehicular environments on the performance of AHS.

Riiser et al. [REV+12] made real-world throughput measurements over 3G networks in various vehicular scenarios while using AHS. With the traces they collected, they created a location-based bandwidth-lookup database to predict the future bandwidth availability for a given location and adjust the video quality accordingly.

Yao et al. [YKHH11] collected bandwidth traces in a real-world vehicular mobility scenario and observed that AHS brings significant advantages in terms of bandwidth efficiency compared to non-adaptive streaming. They also investigated the effect of AHS segment size and the client buffer size, and provided recommendations.

Müller et al. [MLT12] evaluated the performances of the most popular proprietary AHS implementations (MSS, ALS, and HDS) in a vehicular scenario and compared them with their own prototype implementation of MPEG-DASH in terms of video quality fluctuations, smooth playback capability, and bandwidth efficiency. They concluded that their own MPEG-DASH implementation can compete with the proprietary implementations and is more promising in terms of maximizing QoE. In a further study [MRL+12], they improved their rate adaptation mechanism from [MLT12] and offered an improved MPEG-DASH implementation. In the same study, they also provided a SVC-based MPEG-DASH solution which achieves a higher average bit rate, and thus a better video quality.

3.2 AHS rate adaptation algorithms

In order to deal with the varying network resources and fluctuations of the available TCP throughput in wireless networks, rate adaptation is used to adapt the video transmission rate, i.e., to select the appropriate video levels. Since AHS, in particular DASH, is a pull-based streaming system, rate adaptation decisions are fully controlled by the client. In the recent years, many AHS client side rate adaptation algorithms have been proposed in the literature.

Liu et al. [LBG11] proposed a simple rate adaptation algorithm based on the measured segment fetch time and evaluated it using a network simulator in the presence of different kinds of background traffic. In their work, they assumed that segments are requested and received sequentially (*serial fetching*). In a further study [LBHG12], Liu et al. considered also the *parallel* segment fetching which enables multiple simultaneous HTTP requests. Moreover, they also considered the buffered media time and sharing of the TCP throughput

between different DASH clients. Miller et al. [MQGW12] proposed a client rate adaptation algorithm which aims at resolving the trade-off between the start-up delay and maximizing the average video quality. Jarnikov et al. [JÖ11] used a Markov decision process to determine the adaptation strategy. They proposed calculating the optimal strategy, which maximizes the viewing quality, off-line for a given distribution of segment download times.

Various control-theoretical approaches have also been considered. De Cicco et al. [CMP11] proposed a feedback control mechanism to control the sending buffer size at the server side. Tian et al. [TL12] used the evolution of the buffered media time as a feedback signal to adjust the video level by adopting a proportional-integral (PI) controller. In a further study [TL13], they proposed an improved algorithm specifically designed for mobile devices using AHS over wireless networks. Zhou et al. [ZLZG14] proposed a proportional-derivative (PD) controller to adapt the video bit rate with high responsiveness and stability.

A purely buffer-based adaptation algorithm was proposed by Huang et al. [HJM13]. They argued that rate adaptation algorithms do not need to estimate the network capacity and should instead adapt the video bit rate by directly observing and controlling the client playback buffer. They justified this argument by referring to their previous work [HHH+12] in which they showed that inaccurate estimates of the TCP throughput can trigger a feedback loop which leads to frequent video quality shifts and low-quality video. Following this argument, Li et al. [LZG+14] approached the rate adaptation problem when multiple AHS clients compete at a network bottleneck, and they showed that the TCP throughput measured by an AHS client can no longer be trusted as a reliable reference for video level selection in this case. Hence, they proposed an algorithm which employs a "probe-and-adapt" principle at the application layer, similar to (but independent from) the TCP congestion protocol.

Finally, a number of QoE-aware rate adaptation algorithms have also been proposed. Mok et al. [MLCC12] proposed a QoE-aware DASH system which takes the effect of the video level transition on QoE into account. Alberti et al. [ART+13] proposed enhancing the DASH adaptation logic by feeding it with perceptual user evaluation based on a QoE monitoring model. They demonstrated an improved capability to select the optimal video level.

In the following, three selected client rate adaptation algorithms by Liu et al. [LBG11], Miller et al. [MQGW12], and Tian et al. [TL13] are explained in detail.

3.2.1 Liu's algorithm

Liu et al. proposed an AHS rate adaptation algorithm based on a smoothed TCP throughput measurement technique [LBG11]. Due to the strong fluctuations of the instantaneous TCP throughput, it first has to be smoothed to accurately measure the network capacity. Thus, in [LBG11], the authors proposed using the segment fetch time (SFT) measured by the client to determine if the bit rate of the current representation matches the available

end-to-end network capacity. SFT denotes a period of time from the time instant of sending a HTTP/GET request for a media segment to the instant of receiving the last bit of the requested media segment.

In a video streaming system, the client can play media smoothly if the transmission rate, which is measured by SFT, is equal to the playing rate, which is measured by the media segment duration (τ). This occurs only if the bit rate of the current segment matches the average TCP throughput. If SFT is higher (lower) than the media segment duration, it means that the average TCP throughput is lower (higher) than the bit rate of the current segment. Hence, the ratio of media segment duration to segment fetch time is used as a metric to determine the suitability of the selected video level to the available average TCP throughput and is denoted by μ:

$$\mu = \frac{\tau}{\text{SFT}} \tag{3.1}$$

The algorithm's pseudocode is provided in Algorithm 1. The proposed rate adaptation algorithm determines the video level of the next media segment each time after receiving a media segment. The algorithm uses a *stepwise* switch up and *aggressive* switch-down method, i.e., it switches up only one video level, but it might switch down several video levels at a time. Thus, it follows a conservative switch-up strategy in order to prevent playback interruptions.

Algorithm 1: Liu's algorithm [LBG11]

Input: μ, ϵ, γ_d, $\beta_{\min}$
Output: $r_{n(t)+1}, t_{\text{idle}}$

if $\mu > 1 + \epsilon$ **then**
  $r_{n(t)+1} \leftarrow r^{\uparrow}_{n(t)}$
else if $\mu < \gamma_d$ **then**
  $r_{n(t)+1} \leftarrow$ first $r_i < \mu \cdot r_{n(t)}$
else
  $r_{n(t)+1} \leftarrow r_{n(t)}$
$t_{\text{idle}} = \max\left\{\beta(t) - \beta_{\min} - \frac{r_{n(t)}}{r_{\min}} \cdot \tau, 0\right\}$

For switch-up decisions, a switch-up factor ϵ is defined as:

$$\epsilon = \max\left\{\frac{r_{i+1} - r_i}{r_i}\right\}, \forall i = \{0, 1, \cdots, N-1\} \tag{3.2}$$

where r_i denotes the bit rate of segment i, and N denotes the highest video level. Switch-down is performed if the following equality is met:

$$\mu > \gamma_d \tag{3.3}$$

where γ_d denotes a pre-defined switch-down threshold. For an aggressive switch-down, the next video level is determined as the first video level (in descending order) r_i that satisfies:

$$r_i < \mu \cdot r_{n(t)} \tag{3.4}$$

where $r_{n(t)}$ denotes the video level of the current segment. The algorithm keeps the same video level, if $\gamma_d < \mu < 1+\epsilon$ (line 6). If the buffered media time is large enough to cover the maximum draining of the buffer during a segment fetch, the algorithm delays the download of the next segment for a certain period of time. This *idle time* between the end of the current segment download and the start of the next segment download is calculated as:

$$t_{\text{idle}} = \max\left\{\beta(t) - \beta_{\min} - \frac{r_{n(t)}}{r_{\min}} \cdot \tau, 0\right\} \tag{3.5}$$

where $r_{\min}$ denotes the minimum video level, $\beta(t)$ denotes the buffered media time, and $\beta_{\min}$ denotes the pre-defined minimum buffered media time, respectively. Idle time method limits the maximum amount of buffered video time, and thus saves bandwidth and memory resources of the receiver.

3.2.2 Miller's algorithm

Miller et al. proposed a AHS rate adaptation algorithm which aims to optimize the viewing experience of the user [MQGW12]. In the following, the system model, the adaptation goals and the developed algorithm are described.

System model It is assumed that the video stream consists of n segments each containing τ seconds of playback time. $\mathcal{R}_f$ denotes the set of representations for a given client. Average bit rate of a representation $r \in \mathcal{R}_f$ is denoted by $\hat{\rho}_r$. The representation with the highest (lowest) bit rate in $\mathcal{R}_f$ is denoted by $r_{\max}$ ($r_{\min}$). $r^{\uparrow} \in \mathcal{R}_f$ ($r^{\downarrow} \in \mathcal{R}_f$) denotes the next higher (lower) video level with respect to r, if $r \neq r_{\max}$ ($r \neq r_{\min}$). The time of the start (end) of the download of the segment i is denoted by t_i^b (t_i^e). It is assumed that $t_1^b = 0$. t_{start} denotes the time when the first bits of the first segment arrive at the client.

Between the arrival and playback time at the client side, the video is locally buffered at the streaming client. $\beta(t)$ denotes the buffer level in seconds of playback time at time t. $\beta_{\min}(t)$ denotes the minimum buffer level observed during a time interval with duration Δ_β, where Δ_β is a configuration parameter.

$$\beta_{\min}(t) = \min\left\{\beta(t') \,\middle|\, t' \in \left[\left\lfloor\frac{t}{\Delta_\beta}\right\rfloor\Delta_\beta, \left\lfloor\frac{t}{\Delta_\beta}\right\rfloor\Delta_\beta + \Delta_\beta\right]\right\} \tag{3.6}$$

where $\lfloor\cdot\rfloor$ denotes the largest previous integer.

For each adaptation decision, the video level of the downloaded segment is denoted by $r_i \in \mathcal{R}_f$. The segment throughput of the segment i is defined as $\tilde{\rho}_i = (\tilde{\rho}_{r_i} \cdot \tau) \,/\, \left(t_i^e - t_i^b\right)$. Hence, the average segment throughput during the time interval $[t_1, t_2]$ is calculated as:

$$\tilde{\rho}(t_1, t_2) = \frac{\sum_{i=1}^{n(t_2)} \tilde{\rho}_i \cdot \left|\left[t_i^b, t_i^e\right] \cap [t_1, t_2]\right|}{\sum_{i=1}^{n(t_2)} \left|\left[t_i^b, t_i^e\right] \cap [t_1, t_2]\right|}$$

where $n(t_2)$ is the number of segments that are fully downloaded until time t_2 and $|[a,b]| = b-a$ is the length of the time interval $[a,b]$. Finally, $\sigma_i = \left(r_i, t_i^b, t_i^e\right)$ denotes a set of parameters related to the download of the segment i.

Adaptation goals The adaptation algorithm has the following optimization goals: (i) avoid playback interruptions due to buffer underruns: $\beta_t > 0$ for all $t > t_{\text{start}}$, (ii) maximize the minimum and average video quality, (iii) minimize the number of video quality shifts, (iv) minimize the start-up time (t_{start}). There is a trade-off between the goals (ii) and (iii) since downloading each segment with the highest possible video level results in frequent changes of playback quality due to large fluctuations of the available TCP throughput [MQGW12]. Also, there is another trade-off between (ii) and (iv) since t_{start} is minimized if the first segment is downloaded at the lowest video level.

Miller et al. assign (i) the highest priority, i.e., aim to avoid buffer underruns at all costs. Furthermore, they introduce a *fast-start* phase to increase the video quality in an aggressive manner at the beginning of the playback. This mitigates the conflict between the goals (ii) and (iv). The algorithm also aims to achieve a balance between the goals (ii) and (iii).

Algorithm description The algorithm's pseudocode is provided in Algorithm 2. The algorithm is invoked at time t, immediately after the download of segment $n(t)$ is completed. The algorithm has two input arguments: (i) information about the available TCP throughput in the past, $\sigma_i = \left(r_i, t_i^b, t_i^e\right)$ and (ii) buffer level, $\beta(t')$, $t' \in [0,t]$. The algorithm has two output arguments: (i) video level of the next segment, $r_{n(t)+1}$ and (ii) the buffer level threshold that is required to start the download of the next segment, B_{delay}. Hence, the download of the segment $n(t)+1$ is not started before the buffer level $\beta(t)$ falls below B_{delay}. The purpose of B_{delay} is to reduce the buffer level when it exceeds a certain value, hence its function is similar to the idle time (t_{idle}) introduced in Liu's algorithm (Section 3.2.1).

Additionally, the algorithm contains several configuration parameters: $0 \leq B_{\text{min}} < B_{\text{low}} < B_{\text{high}}$, $\Delta_\beta > 0$, $\Delta_t > 0$, $0 < \alpha_1, \cdots, \alpha_5 \leq 1$.

The algorithm always downloads the first segment at the lowest video level. This minimizes t_{start}, i.e. the waiting time of the user after requesting to watch the video stream. The disadvantage of this approach is that the user has to watch the first few segments at a low quality. To mitigate this, the algorithm starts in the fast-start phase and aggressively switches up to quickly improve the viewing experience of the user. In the following, firstly, the operation of the algorithm after the fast-start phase (lines 20-35) is described.

Algorithm 2: MILLER'S ALGORITHM [MQGW12]

Input: $(\sigma_i)_{i=1,\cdots,n(t)}$, $\beta(t)$
Output: $r_{n(t)+1}$, B_{delay}

1 runningFastStart := true
2 $B_{\text{delay}} = 0$
3 $r_{n(t)+1} = r_{n(t)}$
4 **if** runningFastStart ...
5 $\quad \wedge\, r_{n(t)} \neq r_{\max}$ ...
6 $\quad \wedge\, \beta_{\min}(t_1) \leq \beta_{\min}(t_2)\, \forall t_1 < t_2 \leq t$ ...
7 $\quad \wedge\, r_{n(t)} \leq \alpha_1 \cdot \tilde{\rho}(t - \Delta_t, t)$ **then**
8 $\quad$ **if** $\beta_t < \beta_{\min}$ **then**
9 $\quad\quad$ **if** $r^{\uparrow}_{n(t)} \leq \alpha_2 \cdot \tilde{\rho}(t - \Delta_t, t)$ **then**
10 $\quad\quad\quad$ $r_{n(t)+1} \leftarrow r^{\uparrow}_{n(t)}$
11 $\quad$ **else if** $\beta_t < \beta_{\text{low}}$ **then**
12 $\quad\quad$ **if** $r^{\uparrow}_{n(t)} \leq \alpha_3 \cdot \tilde{\rho}(t - \Delta_t, t)$ **then**
13 $\quad\quad\quad$ $r_{n(t)+1} \leftarrow r^{\uparrow}_{n(t)}$
14 $\quad$ **else**
15 $\quad\quad$ **if** $r^{\uparrow}_{n(t)} \leq \alpha_4 \cdot \tilde{\rho}(t - \Delta_t, t)$ **then**
16 $\quad\quad\quad$ $r_{n(t)+1} \leftarrow r^{\uparrow}_{n(t)}$
17 $\quad\quad$ **if** $\beta_t > \beta_{\text{high}}$ **then**
18 $\quad\quad\quad$ $B_{\text{delay}} \leftarrow B_{\text{high}} - \tau$
19 **else**
20 $\quad$ runningFastStart := false
21 $\quad$ **if** $\beta_t < \beta_{\min}$ **then**
22 $\quad\quad$ $r_{n(t)+1} \leftarrow r_{\min}$
23 $\quad$ **else if** $\beta_t < \beta_{\text{low}}$ **then**
24 $\quad\quad$ **if** $r_{n(t)} \neq r_{\min} \wedge r_{n(t)} \geq \tilde{\rho}_{n(t)}$ **then**
25 $\quad\quad\quad$ $r_{n(t)+1} \leftarrow r^{\downarrow}_{n(t)}$
26 $\quad$ **else if** $\beta_t < \beta_{\text{high}}$ **then**
27 $\quad\quad$ **if** $r_{n(t)} = r_{\max}$
28 $\quad\quad\quad\quad \vee\, r^{\uparrow}_{n(t)} \geq \alpha_5 \cdot \tilde{\rho}(t - \Delta_t, t)$ **then**
29 $\quad\quad\quad$ $B_{\text{delay}} \leftarrow \max(\beta(t) - \tau, B_{\text{opt}})$
30 $\quad$ **else**
31 $\quad\quad$ **if** $r_{n(t)} = r_{\max}$
32 $\quad\quad\quad\quad \vee\, r^{\uparrow}_{n(t)} \geq \alpha_5 \cdot \tilde{\rho}(t - \Delta_t, t)$ **then**
33 $\quad\quad\quad$ $B_{\text{delay}} \leftarrow \max(\beta(t) - \tau, B_{\text{opt}})$
34 $\quad\quad$ **else**
35 $\quad\quad\quad$ $r_{n(t)+1} \leftarrow r^{\uparrow}_{n(t)}$

Based on the buffer-related parameters of the algorithm ($0 \leq B_{\min} < B_{\text{low}} < B_{\text{high}}$), a target interval for the buffer level is defined as $\mathcal{B}_{\text{tar}} = [B_{\text{low}}, B_{\text{high}}]$. The middle of the target interval is denoted by $B_{\text{opt}} = 0.5(B_{\text{low}} + B_{\text{high}})$. The algorithm is designed to keep the buffer level close to B_{opt}.

Whenever the buffer level falls below $B_{\min}$, the algorithm immediately switches to the lowest available video level, $r_{\min}$ (lines 21-22). There is a high probability of a buffer underrun if $\beta(t) < B_{\min}$. Hence, in this case, the algorithm tries to avoid playback interruptions by switching down aggressively.

If the buffer level decreases below B_{low} (but is still higher than $B_{\min}$), the algorithm reacts by selecting a lower video level provided that the segment throughput of the last downloaded segment ($\tilde{\rho}_{n(t)}$) is smaller than the current video level ($r_{n(t)}$) (lines 23-25). This condition prevents the algorithm from changing to a lower video level, if the available throughput is high and the buffer level is increasing.

The algorithm never changes the current video level while $\beta(t) \in \mathcal{B}_{\text{tar}}$ (lines 26-28). Hence, it does not react to short-term quality fluctuations since switching rapidly back and forth between different video levels decreases the viewing experience [NEE+11a]. Sensitivity of the algorithm to rapid throughput fluctuations is configured by adjusting the size of

the target interval $\mathcal{B}_{\text{tar}}$. Furthermore, whenever the buffer level is in $[B_{\text{opt}}, B_{\text{high}}]$, yet the conditions for switching to a higher video level is not fulfilled, download of the next segment is delayed by B_{delay} until the buffer level falls back to B_{opt} (line 29).

Finally, the buffer level might exceed B_{high} due to an increase of the available throughput. In this case, there are two options: staying at the current video level and delaying the download of the next segment (line 33), or selecting the next higher video level (line 35). The available throughput measured in the last Δ_t seconds determines the decision. If the next higher video level is above a certain percentage of the throughput (line 32), the current video level is retained and the request is delayed in order to bring the buffer level back to the target interval. Otherwise, the algorithm switches up to the next higher level. α_5 is a safety margin that is used to increase the robustness of the algorithm to measurement uncertainties. The algorithm acts more conservatively in terms of switching up, if $\alpha_5 < 1$ (as opposed to $\alpha_5 = 1$)

In the fast-start phase (lines 4-18), the subsequent segment is requested at the next higher video level ($r^{\uparrow}_{n(t)}$) as long as its bit rate is below a certain percentage (determined by $\{\alpha_1, \cdots, \alpha_5\}$) of the average throughput measured over the last Δ_t seconds. The considered percentage of the throughput that sets this threshold is determined by the current buffer level. For $\beta(t) < B_{\text{min}}$, the algorithm switches to the next higher level if $r^{\uparrow}_{n(t)} \leq \alpha_2 \cdot \tilde{\rho}(t - \Delta_t, t)$. For $\beta(t) \in [B_{\text{min}}, B_{\text{low}}]$, $\alpha_3 \geq \alpha_2$; and for $\beta(t) \geq B_{\text{low}}$, $\alpha_4 \geq \alpha_3$ is used. Hence, the algorithm selects the video levels more aggressively as the buffer level increases. The fast-start phase is terminated if one of the following conditions is not satisfied: (i) $r_{n(t)} \neq r_{\max}$ (line 5), (ii) $\beta_{\min}(t_1) \leq \beta_{\min}(t_2)\, \forall t_1 < t_2 \leq t$ (line 6), or (iii) $r_{n(t)} \leq \alpha_1 \cdot \tilde{\rho}(t - \Delta_t, t)$ (line 7). Condition (i) terminates the fast-start phase if the highest video level is reached. Condition (ii) forces the algorithm to switch to the normal mode of operation if the minimum buffer level observed over a time interval is not increasing. Finally, condition (iii) forces the algorithm to quit the fast start mode when the selected video level is too close to the measured available throughput.

3.2.3 Tian's algorithm

Tian et al. proposed an AHS rate adaptation algorithm which calculates a target bit rate for the next segment based on the current buffered video time, the recent TCP throughput history and video levels of the previous segments [TL13]. The main goal of the algorithm is to achieve high perceptual video quality without suffering from playback interruptions. In the following, first their system model is given, then the algorithm is described.

System model In their previous study [TL12], Tian et al. showed that simple history-based TCP prediction has a very good accuracy in the context of DASH, as compared to other more complex prediction algorithms. Hence, they use the average TCP throughput $\bar{T}(k)$ of each segment from $k - N$ to $k - 1$ to estimate the average TCP throughput $\hat{T}(k)$

of the current segment k. The average TCP throughput for a segment i is given as:

$$\bar{T}(i) = \frac{\nu(i)\tau}{t_i^e - t_i^b} \tag{3.7}$$

where $\nu(i)$ is the video level of the ith downloaded segment, τ is the segment duration, and t_i^b (t_i^e) is the download start (end) time of the segment i. Thus, the estimated TCP throughput while downloading the current segment k is given as:

$$\hat{T}(k) = \frac{1}{N} \sum_{i=k-N}^{k-1} \bar{T}(i) \tag{3.8}$$

where N is the length of the considered TCP throughput history.

However, the target video bit rate is not directly set to $\hat{T}(k)$ since this might lead to playout interruptions if the estimate is overly aggressive [TL12]. Instead, a dynamic rate margin is employed as a safety measure to account for the instability of the TCP throughput. To this end, first a *smoothing index* SI is defined to quantify the consistency of the TCP throughput:

$$\mathrm{SI}(k) = \frac{1}{N-1} \sum_{k-N}^{k-1} \frac{|\bar{T}(i) - \bar{T}(i-1)|}{\bar{T}(i)} \tag{3.9}$$

The dynamic rate margin is then calculated as a function of SI by using a logistic function:

$$M(k) = 1 - \frac{25 + 70\mathrm{e}^{\mathrm{SI}(k)}}{100\mathrm{e}^{\mathrm{SI}(k)}} \tag{3.10}$$

Thus, the target video bit rate is set to $\tilde{\nu}_{\mathrm{tar}}(k) = \hat{T}(k)(1 - M(k))$. Considering that only a number of finite video levels exist in an AHS system, the target video bit rate is quantized to get the largest affordable video level:

$$\tilde{\nu}(k) = Q(\tilde{\nu}_{\mathrm{tar}}(k)) \triangleq \max_{\{\nu_i : \nu_i \leq \tilde{\nu}(k)\}} \nu_i \tag{3.11}$$

Aside from $\tilde{\nu}(k)$, the algorithm also uses the buffer level measured at the end of the download of each segment (denoted by $q(k)$) as a secondary measure for rate adaptation decisions.

Algorithm description The algorithm's pseudocode is provided in Algorithm 3.

Algorithm 3: TIAN'S ALGORITHM [TL13]

Input: $\hat{T}(k)$, $q(k)$
Output: $v(k)$, t_{idle}

```
1  if q(k) < q_thr/2 then
2  |  ν(k) = Q(T̄(k−1)(1 − M(k)))
3  |  return
4  else
5  |  ν̂(k) = Q(T̂(k)(1 − M(k)))
6  |  if ν̂(k) > ν(k − 1) then
7  |  |  Counter++
8  |  |  if Counter > m then
9  |  |  |  ν(k) ← ν̂(k)
10 |  |  |  Counter ← 0
11 |  |  else
12 |  |  |  ν(k) ← ν(k − 1)
13 |  else
14 |  |  Counter ← 0
15 |  |  ν(k) ← ν(k − 1)
16 t_idle = max{q(k) − q_cap, 0}
```

The algorithm switches down only if the buffered video time falls below the half of a predefined threshold q_{thr} (lines 1-3). In this case, the target video level is determined by the average TCP throughput of the previous segment discounted by the dynamic rate margin.

If the buffer level is higher than $q_{\text{thr}}/2$, the algorithm might switch up or remain at the same level depending on the estimated TCP throughput $\hat{T}(k)$. In the case of a switch-up, the algorithm takes a conservative approach. If $\hat{\nu}(k)$ is higher than or equal to the video level of the previous segment $\nu(k-1)$, the algorithm waits for m consecutive segments before switching up, during which $\hat{\nu}(k)$ should consistently be higher than $\nu(k-1)$ (lines 7-10). Otherwise, the waiting process is restarted, and the current video level is retained (line 12). By employing a waiting period, the algorithm makes sure that the actual TCP throughput can indeed support a higher video level. The number of consecutive segments to wait, denoted by m, is called the *smoothing parameter* since it determines the trade-off between the smoothness and adaptivity of the algorithm. If m is large, TCP throughput fluctuations can be smoothed more effectively, but the algorithm reacts slowly to available throughput increases. With smaller m, the algorithm can quickly switch up to a higher video level, but TCP throughput estimation errors might lead to buffer underruns. Therefore, the algorithm adjusts m dynamically based on the TCP throughput increase trend given by

$\Delta\bar{T}(k) = \bar{T}(k) - \bar{T}(k-1)$. The smoothing parameter m is calculated as:

$$m(k) = \begin{cases} 3 & \text{if } \Delta\bar{T}(k) \in \left[0.4\bar{T}(k-1), \bar{T}(k-1)\right] \\ 8 & \text{if } \Delta\bar{T}(k) \in \left[0.2\bar{T}(k-1), 0.4\bar{T}(k-1)\right] \\ 15 & \text{if } \Delta\bar{T}(k) \in \left[0, 0.2\bar{T}(k-1)\right] \\ 20 & \text{otherwise} \end{cases} \tag{3.12}$$

Thus, the waiting period is shortened when there is an obvious TCP throughput increase trend. To reduce the influence of outliers, the algorithm averages three consecutive calculations from Equation 3.12 and uses the average as the smoothing parameter.

On the other hand, if $\hat{\nu}(k)$ is lower than $\nu(k-1)$, no video level change occurs (line 15). Finally, whenever the buffered video time $q(k)$ exceeds the buffer cap q_{cap}, the algorithm delays the download of the next segment by $q(k) - q_{\text{cap}}$ seconds (line 16).

3.3 Video quality assessment

In multimedia applications, there are two ways to assess video quality [IT13]. In principle, for a reliable assessment, subjective tests must be performed, and user satisfaction must be evaluated with metrics based on the user ratings. However, it is also possible and usually more practical to estimate the video quality by means of objective VQMs based on quality estimation models [ETS10]. Subjective testing is more expensive and needs more effort since it requires human subjects. On the other hand, automatic perceptual quality estimation based on objective VQMs is generally much faster and cheaper. However, reliability of the perceptual quality estimation depends on the accuracy of these models. In the following, a brief overview of subjective video quality assessment is given, and state-of-the-art objective VQMs are reviewed.

3.3.1 Subjective quality assessment

Subjective video quality assessment methods are more accurate than the objective ones (when conducted properly) since they directly take the properties of HVS into account [LE12]. Furthermore, they serve as a reference for the performance evaluation of objective VQMs, i.e., the perceptual quality estimated by the objective models is always compared with the perceptual quality measured by the subjective tests to determine the degree of accuracy of the objective models [CSRK11]. ITU recommends a variety of subjective test methods depending on the particular assessment problems. These problems might be, for example, measuring the quality of a system relative to a reference, or comparing the quality of alternative systems when no reference is available [ITU12]. For all of these methods, the obtained perceptual quality ratings are averaged to obtain the mean opinion score (MOS).

Certain subjective test methods use the differential mean opinion score (DMOS) which is computed by subtracting the score assigned by the user to the processed video sequence (PVS) from the score assigned by the same user to the corresponding source video sequence (SRC) [CSRK11].

3.3.2 Objective video quality metrics

In Section 2.1.1, it has been discussed that quality metrics which only consider spatial quality impairments (e.g. PSNR, SSIM) might not correlate well with the measured perceptual quality (by subjective assessment) in video streaming applications. In adaptive bit rate streaming, the best possible trade-off between spatial and temporal quality must be determined such that the user satisfaction is maximized [LMS+14]. Therefore, *spatio-temporal* quality metrics are required that take both spatial and temporal impairments of the original video into consideration.

In [FSWV07], Feghali et al. proposed a VQM that considers both frame rate and QP changes. The metric is based on a weighted sum of PSNR values and frame rate reduction, where the weight is determined by a motion measure. However, it does not account for the content-dependency of PSNR, thus it cannot provide accurate quality predictions across different video contents. Also, the motion measure depends on the motion estimation scheme used in the video codec, i.e., the metric is codec-dependent.

In [OMW09], Ou et al. proposed a VQM which is the product of a PSNR-based spatial quality metric and a temporal correction factor. Although this VQM only contains two parameters, they need to be determined for each video individually. Hence, the metric is content-dependent and not suitable for an automatic system. Ou et al. extended their VQM in [OMLW11] by estimating the metric parameters using content features. However, for each source video four spatial and temporal parameters need to be calculated, and calculation of these parameters are computationally complex.

Peng and Steinbach [PS11] proposed another spatio-temporal VQM which has a lower computational complexity than the metric proposed in [OMLW11] and is neither content- nor codec-dependent. The authors named their metric spatio-temporal video quality metric (STVQM). In [LMS+14], Lottermann et al. considered uplink video streaming in a vehicular environment and extended STVQM by developing a low-complexity model to estimate TA and SA based on contextual information provided by vehicle sensors. Accordingly, they named their metric contextual STVQM (CSTVQM).

Machado [Mac13] performed a subjective test to measure the MOS for various automotive videos and investigated the performance of the VQMs from [FSWV07], [OMW09], and [PS11] as well as the PSNR model in terms of Pearson correlation (PC) and root-mean-square error (RMSE) with the measured MOS values. The results show that the predicted MOS values from [OMW09] and [PS11] correlate significantly better with the measured MOS values than [FSWV07] and the PSNR model.

In the following, STVQM model [PS11] is briefly presented.

STVQM

STVQM depends on PSNR, frame rate, and two standard video activity measures[1], TA and SA which need to be extracted from the uncompressed video. STVQM can be applied in rate control to trade-off between temporal and spatial quality such that the perceptual quality is maximized [PS11].

In [PS11], initially a subjective test is conducted with three SRCs with different spatial and temporal characteristics [SR12] (*Mother&Daughter*, *Foreman*, and *Football*) to identify the effects of frame rate and PSNR variations on the video quality. Each SRC is temporally downsampled to four different temporal quality levels, and each temporal quality level is encoded at three different spatial quality levels. After obtaining the DMOS scores, it is first investigated how the perceived spatial quality changes as a function of PSNR at full frame rate (30 fps). Secondly, impact of temporal impairment on the overall perceived quality is investigated. Spatial video quality is modeled by a logistic function:

$$\mathrm{SVQM} = \frac{100}{1 + \mathrm{e}^{-(\mathrm{PSNR}+\omega_s \cdot \mathrm{SA}+\omega_t \cdot \mathrm{TA}-\mu)/s}} \tag{3.13}$$

where the constants ω_s, ω_t, μ, and s are determined by a least-square non-linear fitting using the subjective data of the considered videos. Similarly, temporal video quality is modeled by the following function:

$$\mathrm{TVQM} = \frac{1 + a \cdot \mathrm{TA}^b}{1 + a \cdot \mathrm{TA}^b \cdot \frac{f_{\max}}{\mathrm{FR}}} \tag{3.14}$$

where FR is the frame rate of the PVS, $f_{\max}$ is the frame rate of the SRC while a and b are model parameters determined by a least-square fitting of the subjective data. Finally, based on the analysis of variance (ANOVA) results which show that an interaction exists between spatial and temporal quality perception, STVQM is calculated as the product of SVQM (3.13) and TVQM (3.14):

$$\mathrm{STVQM} = \mathrm{SVQM} \cdot \mathrm{TVQM} \tag{3.15}$$

For validation, performance of STVQM is evaluated over the entire dataset, and it is shown that the predicted DMOS values of STVQM correlate very well with the subjective test ratings resulting in a PC of 0.994 and an RMSE of 2.66%.

[1] See Section 2.1.2 for an explanation of content-dependent video parameters.

3.4 Vehicle on-board and off-board architecture

The focus of the present thesis is to upstream a video from the front-facing camera of a vehicle to an off-board entity. For this purpose, vehicular on-board components need to be defined and an end-to-end network architecture needs to be constructed. In the following, a proposed network architecture and streaming-related on-board components are described.

3.4.1 End-to-end network architecture

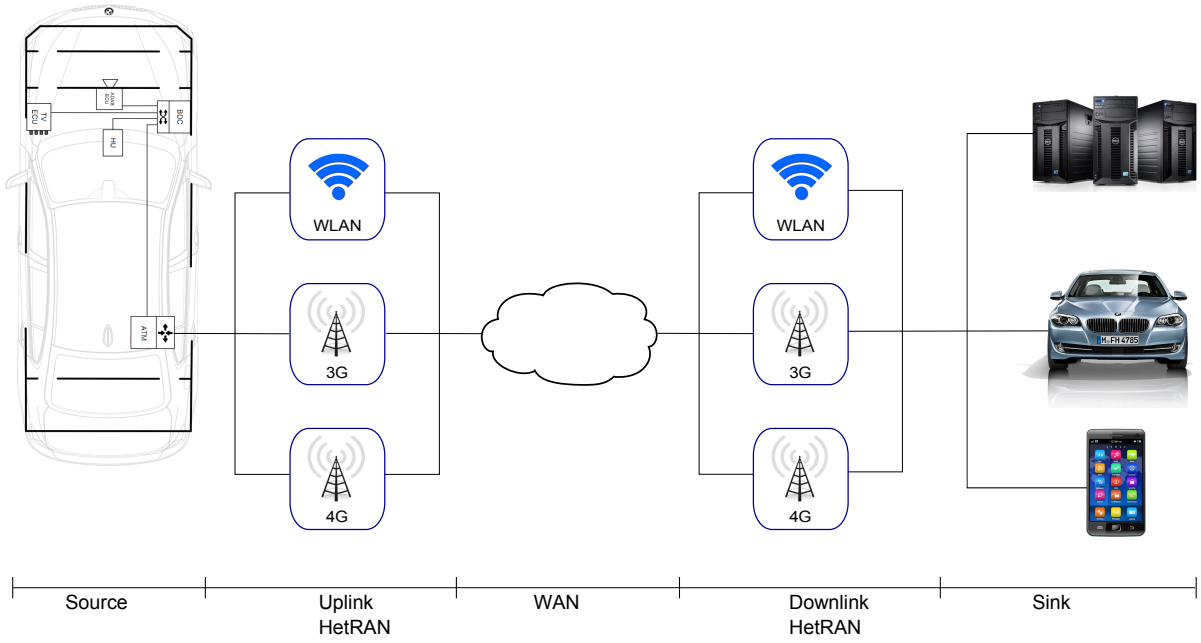


Figure 3.3: Considered end-to-end network architecture for vehicular DASH.

Figure 3.3 illustrates the considered end-to-end network architecture for a vehicular uplink video streaming scenario [LS13]. The video source is a vehicle that streams the video of its front-facing camera to a sink which can be another vehicle, an off-board server, or a consumer electronics device such as a smartphone. Between the video source and the sink, cascaded networks exist which consist of the uplink RAN of the video source, a wide area network (WAN) infrastructure, and the downlink RAN of the sink [LGSS]. The video can be streamed over modern mobile RANs such as HSPA networks (3G) or LTE networks (4G), or over wireless local area networks (WLANs). The heterogeneous RANs (HetRANs), which are used in the uplink direction from the vehicle or in the downlink direction to the sink, do not necessarily have to be of the same kind. Among the cascaded networks between the video source and sink, the network with the lowest transmission capacity constitutes a bottleneck for the overall end-to-end network capacity [LS13]. Hence, the mean available

end-to-end network throughput is given as:

$$\overline{W}_{\text{E2E}} = \min\left\{\overline{W}_{\text{UL,HetRAN}}, \overline{W}_{\text{DL,HetRAN}}, \overline{W}_{\text{WAN}}\right\} \tag{3.16}$$

where $\overline{W}_{\text{UL,HetRAN}}$, $\overline{W}_{\text{DL,HetRAN}}$, $\overline{W}_{\text{WAN}}$ denote the mean available throughput of the uplink HetRAN, the mean available throughput of the downlink HetRAN, and the mean available throughput of WAN, respectively.

3.4.2 On-board vehicular architecture

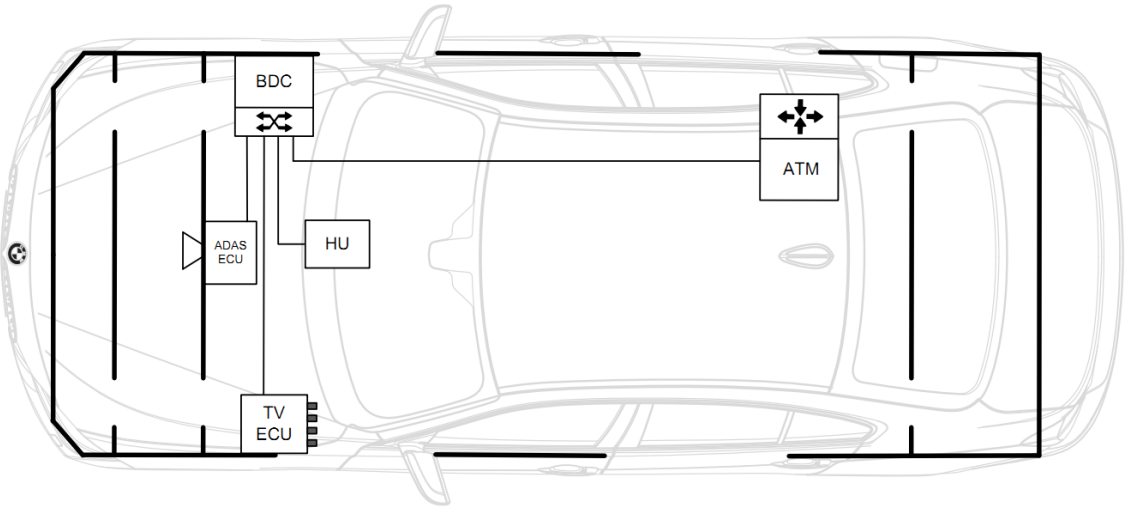


Figure 3.4: Considered on-board architecture for vehicular DASH [LS13].

Modern premium cars contain several electronics control units (ECUs) interconnected by various automotive-specific network technologies. For video-streaming applications, only a subset of the available ECUs are used. Figure 3.4 display the required ECUs. However, the limited transmission capacities of these networks and the inflexible network architecture no longer satisfy the demands of the resource-hungry applications such as video streaming [Rah09]. In the recent years, application of an all IP-over-Ethernet in-car network has been considered since the Ethernet technology offers enough capacity for in-vehicle transmission of multimedia data [Rah09]. Hence, the considered on-board architecture assumes that all ECUs are connected via Ethernet [LS13]. In the proposed architecture, the raw video stream is gathered and processed by the Advanced Driver Assistance Services (ADAS) ECU. ADAS ECU is the main processing unit for the video from the front-facing camera of the vehicle whereas the Top View ECU processes the videos from the four fisheye cameras mounted around the vehicle to create a top-view perspective. Figure 3.5 illustrates the different camera systems in a modern premium car. The Head Unit (HU) performs central processing and control of all the other applications inside the vehicle, and also displays

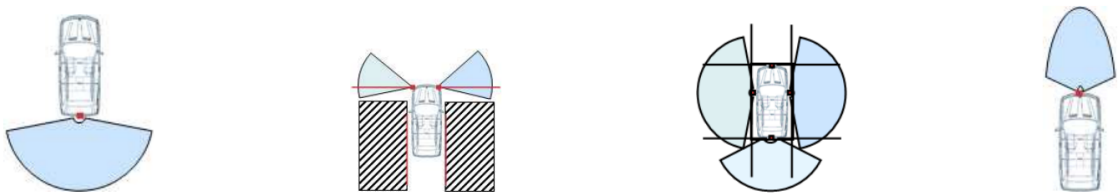


Figure 3.5: Different in-car camera systems: rear-view, side-view, top-view, front-view (left to right) [Rah09].

the contents on the Central Information Display (CID) in the car. The Body Domain Controller (BDC) routes the traffic of the different ECUs over different domains. Finally, an Advanced Telecommunications Module (ATM) enables some ECUs in the vehicle to connect to IP-based wireless RANs. Due to the heterogeneous nature of those RANs, the ATM requires a reconfigurable wireless receiver design which satisfies the different frequency requirements [HFBS09].

Chapter 4

TCP performance measurements of HSPA and LTE networks in vehicular environments

In this chapter, measurements of uplink and downlink TCP throughput in an automotive environment in real HSPA and LTE networks are described. Repeated measurements were performed in urban and highway scenarios to determine the network performance and to create a geo-referenced database. The obtained database can be used to evaluate the performance of various rate adaptation algorithms. Section 4.1 presents the measurement procedure and the post-processing steps. Section 4.2 presents the network performance results for single RAN (HSPA or LTE) and inter-RAN handover scenarios.

4.1 Experimental settings

4.1.1 Measurement setup

The TCP uplink and downlink performance of RANs with an off-board server was recorded in an automotive environment along selected highway and urban routes in Munich. The experiments were performed in the real HSPA and LTE networks of Telefónica. HSPA and LTE coverage maps of Telefónica in Munich were analyzed to determine potential tracks. The tracks which mostly provide uninterrupted HSPA and LTE coverage were preferred. As urban route, a 4.3 km long track in the Munich central area between Georg-Brauchle-Ring and Nymphenburg was selected (Figure 4.1a). As highway route, a 12.6 km long track on A8 highway between Graßlfing and Odelzhausen was selected (Figure 4.1b).

The experimental setup is illustrated in Figure 4.2. The measurement hardware consists of a ZTE MF821 heterogeneous RAN modem connected to the ATM of the vehicle and an

external automotive antenna system with two printed circuit board LTE MIMO antennas, mounted on the vehicle rooftop [ETKM13]. The gain difference of the MIMO antennas used in the experiments leads to a direction-dependent performance of the antenna system in the vicinity of the LTE base station [ETKM13]. Hence, to ensure reliable performance measurements, one driving direction was reserved for the uplink and the other for the downlink measurements. On the ATM side, a custom measurement tool implemented in the robot operating system (ROS) was installed to collect the throughput measurements. To ensure statistical reliability, the experiments were repeated nine times for each network (HSPA or LTE) and each scenario (highway or urban), both in uplink and downlink directions, respectively.

All uplink measurements started at Georg-Brauchle-Ring and ended at Nymphenburg for the urban scenario. The experiments had an average duration of 504 s with an average velocity of 30 km/h. All uplink measurements started at Graßlfing and ended at Odelzhausen for the highway scenario. The experiments had an average duration of 325 s with an average velocity of 140 km/h. All downlink measurements were performed in the opposite directions, i.e., from Nymphenburg to Georg-Brauchle-Ring in the urban scenario and from Odelzhausen to Graßlfing in the highway scenario.

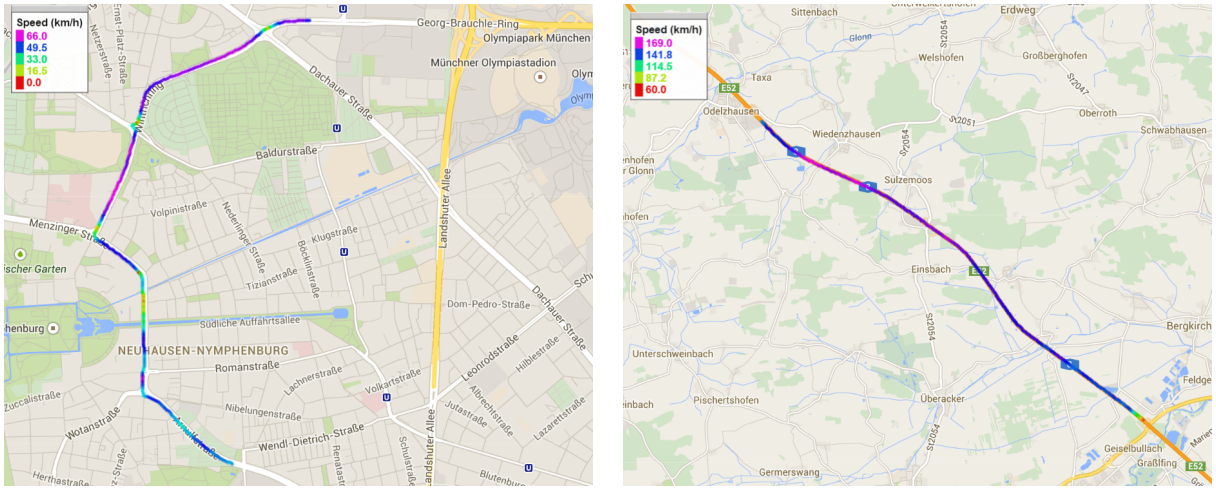
Data points were recorded at a frequency of 10 Hz during each uplink (downlink) measurement. Each data point contains:

1. Current time as Unix timestamp (number of seconds since 1970-01-01 00:00 UTC)
2. Geographical position (latitude and longitude) obtained by the GPS receiver of the vehicle
3. Velocity of the vehicle
4. Uplink (downlink) TCP throughput

The velocity of the vehicle as a function of position during one of the uplink measurements is shown in Figure 4.1 for urban and highway scenarios, respectively. In the uplink measurements *iperf* [TQD+05] was used to continuously upload packets to a fixed server in the Internet. For the downlink measurements, *wget* [Nik95] was used to continuously download a large file from a fixed server in the Internet. No bandwidth throttling was applied to artificially limit the TCP throughput.

4.1.2 Post-processing

Offline post-processing was applied to the recorded network performance traces to prepare the data for further analysis. The following post-processing steps were taken: First, every 10 data points were averaged to obtain 1-second averages. Thus, the number of data points was reduced tenfold, and a smoothed TCP throughput was obtained. Secondly, the



(a) Urban route (b) Highway route

Figure 4.1: Selected routes for the TCP throughput measurements. Coloured roads show the velocity of the vehicle during one uplink measurement.

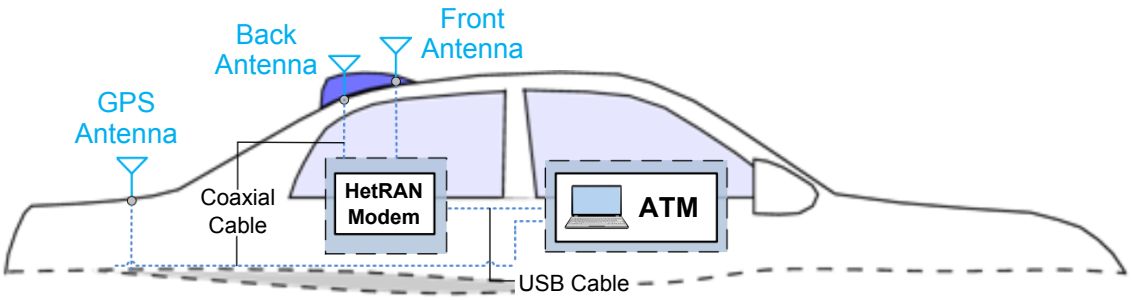


Figure 4.2: Experimental setup (adapted from [ETKM13]).

recorded GPS coordinates were used to determine the distance of the vehicle from its starting position at each data point. Due to the varying intra-session velocities, the recorded data points were not equidistant. Hence, they were re-sampled to obtain equidistant data points, and the measured TCP throughput values were linearly interpolated accordingly. The interpolated and original data point sets have a PC > 0.99. Therefore, it was concluded that the interpolated data set could be used for the rest of the analysis. Thus, it was assumed that the vehicle maintains a constant velocity throughout the measurements and collects equidistant data points. Finally, all the post-processed network performance traces were collected in a prototypical geo-referenced bandwidth database.

For further analysis, an artificial scenario was created where the mobile video source connects to different heterogeneous RANs during an uplink streaming process. Therefore, periodic handovers were introduced that occur every 60 s, and the geo-referenced LTE and HSPA traces obtained from respective uplink TCP throughput measurements were piecewise combined. Thus, nine traces representing handover scenarios between LTE and HSPA networks were generated. Figure 4.3 illustrates one created handover trace.

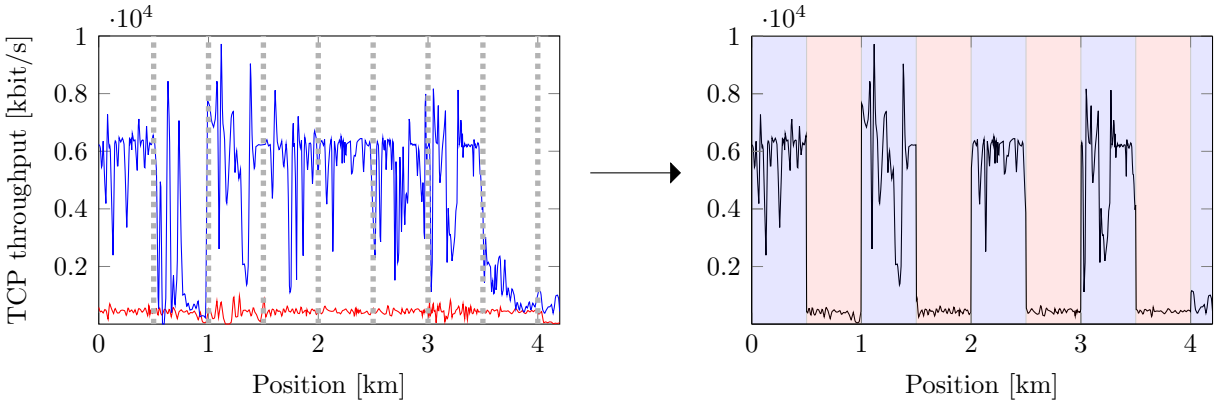


Figure 4.3: Creation of a handover trace (——) from a HSPA (——) and LTE (——) trace. Dotted lines (·····) show the points where the handovers are introduced.

4.2 Network performance results

To evaluate the performance of the HSPA and LTE networks, the mean and standard deviation of the measured TCP throughput of each scenario were calculated. For the single RAN scenarios, where only one of the HSPA or LTE networks is considered, the measured real network traces were used. For the handover scenarios, the artificially created handover traces (Figure 4.3) were used.

4.2.1 Single RAN scenarios

Figure 4.4 and 4.5 show the mean and standard deviation of the TCP throughput over time for uplink and downlink HSPA measurements, respectively. Table 4.1 shows the mean and standard deviation of the mean TCP throughput for HSPA urban and highway scenarios. The results show that the uplink and downlink TCP performances offer a mean value around 400kbit/s for both scenarios.

Figure 4.6 and 4.7 show the mean and standard deviation of the TCP throughput over time for uplink and downlink LTE measurements, respectively. The figures indicate that there are significant TCP throughput differences between urban and highway scenarios both in uplink and downlink. Figure 4.6 shows that the mean uplink TCP throughput of the urban scenario is more than ten times higher than that of the highway scenario. Moreover, the TCP throughput fluctuations are much larger in the urban scenario compared to the highway scenario. Figure 4.7 shows that similar TCP throughput differences between urban and highway scenarios are also present for the downlink measurements. Additionally, for the urban scenario, the mean downlink TCP throughput is significantly higher than the mean uplink TCP throughput. However, the downlink TCP throughput fluctuates much less than the uplink TCP throughput. Table 4.2 shows the mean and standard deviation of the mean TCP throughput for LTE urban and highway scenarios.

Figures 4.4–4.7 and Table 4.1–4.2 demonstrate that the LTE Urban scenario presents a greater challenge to rate adaptation algorithms than the other scenarios due to significantly higher variations of TCP throughput.

Table 4.1: Mean and standard deviation of the measured mean TCP throughput over nine HSPA traces.

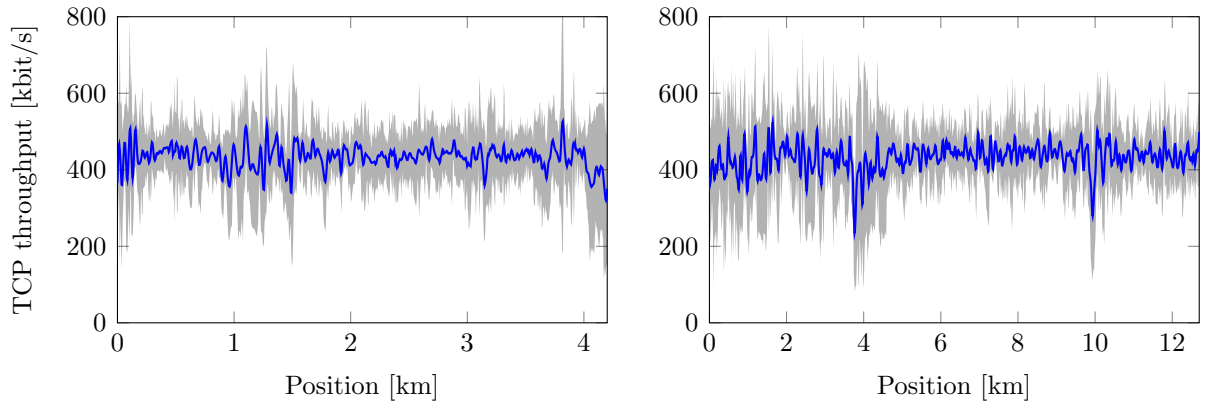
	Uplink [kbit/s]	Downlink [kbit/s]
Urban	433.4 ± 39.1	374.1 ± 18.6
Highway	431.2 ± 40.6	373.5 ± 26.1

Table 4.2: Mean and standard deviation of the measured mean TCP throughput over nine LTE traces.

	Uplink [kbit/s]	Downlink [kbit/s]
Urban	4632.3 ± 1776.4	6771.1 ± 423.9
Highway	416.0 ± 91.0	358.2 ± 48.9

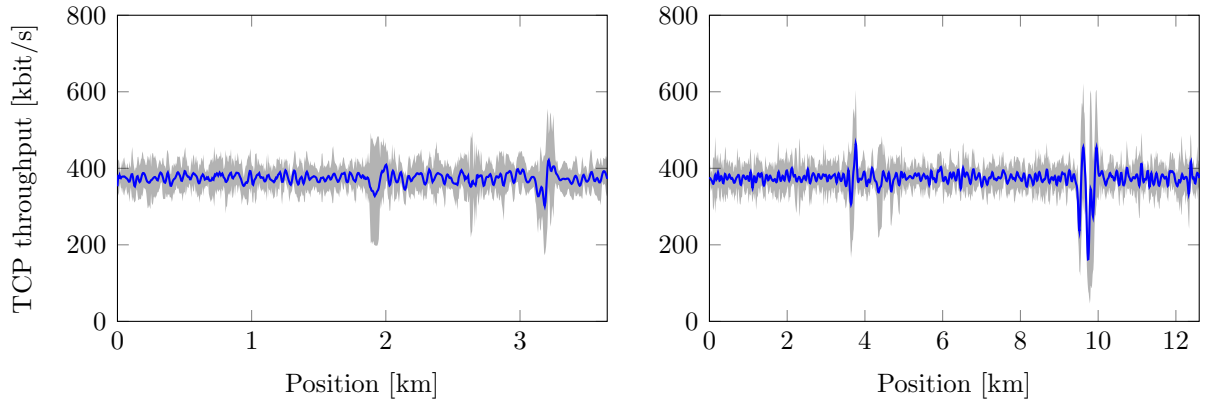
4.2.2 Handover scenarios

Figure 4.8 shows the mean and standard deviation of the TCP throughput over time for uplink handover scenarios. The TCP throughput fluctuates significantly during a handover



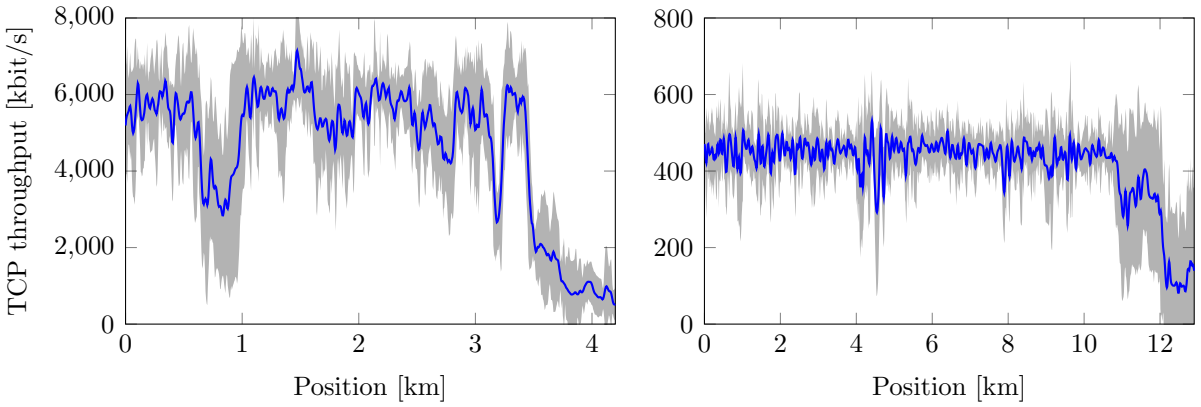
(a) Urban scenario (b) Highway scenario

Figure 4.4: Mean (——) and standard deviation (▬) of the measured uplink TCP throughput over nine HSPA traces.



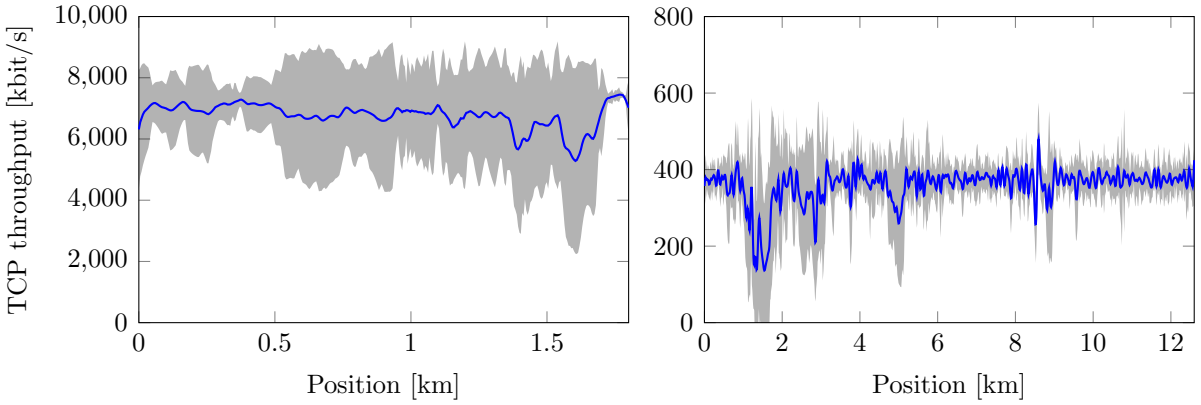
(a) Urban scenario (b) Highway scenario

Figure 4.5: Mean (——) and standard deviation (▬) of the measured downlink TCP throughput over nine HSPA traces.



(a) Urban scenario (b) Highway scenario

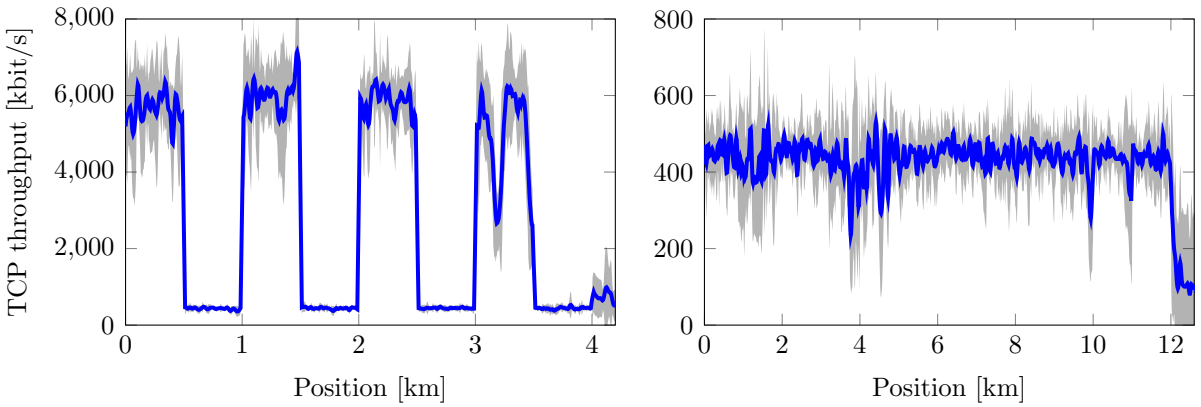
Figure 4.6: Mean (——) and standard deviation (▬) of the measured uplink TCP throughput over nine LTE traces.



(a) Urban scenario (b) Highway scenario

Figure 4.7: Mean (——) and standard deviation (▬) of the measured downlink TCP throughput over nine LTE traces.

in the urban scenario (Figure 4.8a) whereas it remains almost at the same level during a handover in the highway scenario (Figure 4.8b). Therefore, the urban scenario presents a significantly greater challenge to rate adaptation algorithms.



(a) Urban scenario

(b) Highway scenario

Figure 4.8: Mean (——) and standard deviation (▬) of the measured uplink TCP throughput over nine handover traces.

Chapter 5

AHS rate adaptation algorithms for uplink streaming in automotive environments

Network performance characteristics of HSPA and LTE networks in an automotive scenario have been presented in Chapter 4. In this chapter, a vehicular DASH architecture is proposed. An automotive scenario is considered in which the video from a front-facing camera of a vehicle is upstreamed to a video portal with an installed standard AHS adaptation algorithm. Also, a simulation framework is developed to evaluate the performance of three client rate adaptation algorithms in terms of user QoE. Section 5.1 presents the proposed vehicular DASH architecture. Section 5.2 compares the performance of three selected state-of-the-art AHS adaptation algorithms within the proposed vehicular streaming architecture.

5.1 Proposed vehicular DASH architecture

5.1.1 System model

A vehicular upstream live video delivery scenario, in which video is streamed over RANs to a video portal, is considered. It is assumed that during the streaming process, a total of n segments are transmitted each containing τ seconds of playback time. Each segment is available in different representations. In the remaining part of this thesis, it is assumed that the representations have the same spatial resolution, contain only 2D video, and differ only in their video quality levels (bit rate of the video segment). Therefore, different representations are denoted briefly as *video levels* in the rest of this chapter.

Figure 5.1 displays the system model for the proposed vehicular DASH architecture. Inside

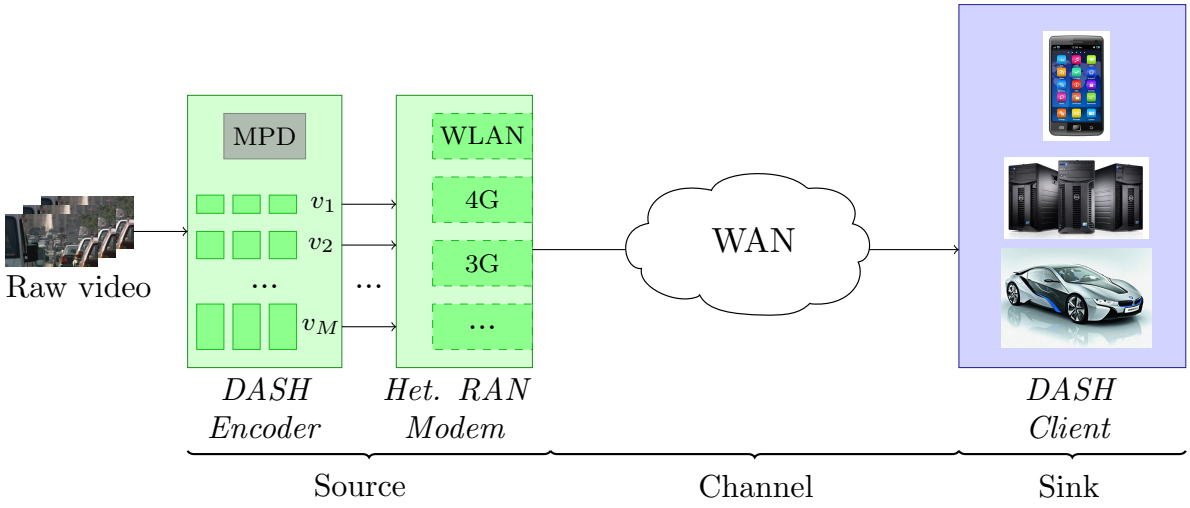


Figure 5.1: Proposed vehicular DASH streaming architecture.

the ADAS ECU of the vehicle, a DASH enabled encoder is set up which encodes the source video at different bit rates simultaneously, splits the encoded videos into segments of length τ, and stores the segments on a locally installed web server. All segments are referenced in an index file (MPD) which is used by the DASH client to request the video streams. A static set $\mathbf{V}$ of M pre-defined video levels is considered[1] . The DASH client adaptively requests segments from the vehicle using a standard AHS adaptation algorithm. The DASH client can be a device like a smartphone or another vehicle which directly consumes the live video, or a back-end server which acts as an intermediate node to store the video for on-demand retrieval. Between the video source and DASH client, cascaded networks exist which consist of the uplink RAN of the video source, the WAN infrastructure and the downlink RAN of the DASH client. It is assumed that the uplink RAN of the video source is the bottleneck for the network performance.

The DASH client is a standard AHS adaptation algorithm which (i) downloads the segments in chronological order, (ii) requests only one video level per download and (iii) downloads the video segments in a non-preemptive manner, i.e., the download of the current segment must be completed before the download of the next segment is started.

Streaming buffer

In video streaming applications, the media client maintains a video buffer to compensate for the mismatches between video download rate and video playback rate. In non-adaptive video streaming, fullness of the video buffer is measured by the size of the buffered video (in bits) which can be mapped to buffered video playback time when divided by the constant video bit rate. However, in AHS, the fullness of video buffer in the client is typically

[1]See Section 5.1.3 for the selection of the pre-defined DASH video levels.

measured in seconds of remaining playback time [TL12]. At any time, buffer may contain segments with many different video levels, and output rate of buffer depends on the video level of the currently played segment. Hence, there is no direct mapping between buffered video size and buffered video time [HJM13].

Buffered video time at time t is denoted by $B(t)$. Since video levels can only be selected on a segment basis, buffer dynamics are considered at time points when the download of a segment is completed. Available uplink TCP throughput for streaming at time t is denoted by $T(t)$. $\nu[k]$ denotes the selected video level, and t_k denotes the completion time of the download of the kth segment. $\overline{T}[k]$ denotes the average TCP throughput for the kth segment and is calculated as:

$$\overline{T}[k] = \frac{\int_{t_{k-1}}^{t_k} T(t)\,\mathrm{d}t}{t_k - t_{k-1}} \tag{5.1}$$

Since each segment contains τ seconds of video, the kth segment has a data size of $\tau\nu[k]$ bits. Therefore, the download completion time of the k-th segment is $t_k = t_{k-1} + \tau\nu[k]/\overline{T}[k]$. Besides, τ seconds of video is added to the buffer between t_{k-1} and t_k. Hence, buffer level at the end of the kth segment is:

$$B(t_k) = \left[B(t_{k-1}) + \tau - \frac{\tau\nu[k]}{\overline{T}[k])}\right]^+ \tag{5.2}$$

where the notation $[x]^+$ means the positive part of x, i.e., $max\{x, 0\}$. Figure 5.2 illustrates the employed streaming buffer model [HJM13].

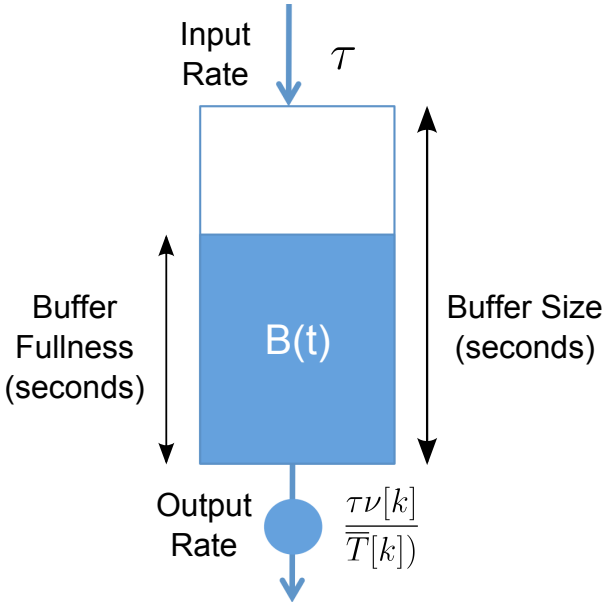


Figure 5.2: Streaming buffer model (adapted from [HJM13]).

5.1.2 Perceptual rate-distortion model

In this section, a perceptual rate-distortion (R-D) model is developed which is used to: 1) determine the DASH levels of the full video level set **V** in a perceptual quality-aware manner, 2) take perceptual quality into account for performance assessment of rate adaptation algorithms.

A video database containing urban and highway road videos is present. As a first step, three 10 s long road videos with diverse spatial and temporal properties are selected from the video database. Figure 5.3 displays the example frames from the selected videos. The videos have a resolution of 1280x720 pixels and a frame rate of 30 frames per second (fps). Table 5.1 shows the calculated TA and SA values for the three selected videos. To quantify the subjective quality for each video level, R-D curves are created with a MOS scale defined in [IR96] which ranges from 1.0 (worst) to 5.0 (best).

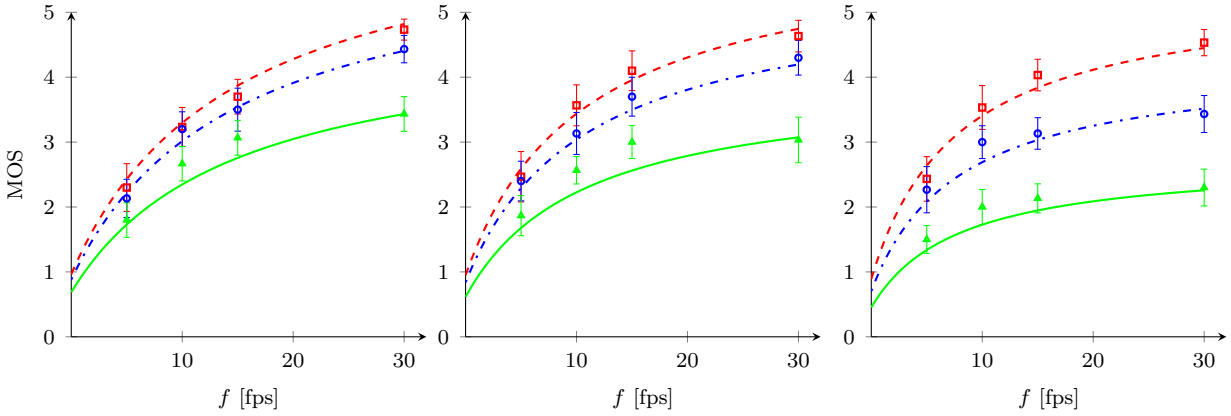
Table 5.1: Measured TA and SA values for the selected road videos.

Video ID	TA	SA
1	26.6	57.2
2	13.3	49.8
3	6.7	28.4

(a) Video 1 (b) Video 2 (c) Video 3

Figure 5.3: Example frames of the selected videos.

STVQM [PS11] is used to model the subjective quality for different spatial and temporal quality impairments. To train the model parameters, first a test based on Subjective Assessment of Multimedia Video Quality (SAMVIQ) [IR08] is conducted to assess the subjective quality. Details of the subjective test methodology are presented in Appendix A. For each of the three uncompressed SRCs, 12 PVSs are generated at four different frame rates (30 fps, 15 fps, 10 fps, 5 fps) and three different PSNR levels (42 dB, 38 dB, 34 dB) in H.264/AVC main profile at a constant QP using x264 [X26]. The screening procedure proposed in [IR08] is used to exclude outliers from the ratings. After the screening procedure,



(a) Video 1 (b) Video 2 (c) Video 3

Figure 5.4: Calculated MOS values based on trained STVQM [PS11]: 34 dB (——), 38 dB (·-·-), 42 dB (- - -); MOS obtained from subjective test: 34 dB (△), 38 dB (○), 42 dB (□) with a 95% confidence interval (CI).

15 out of 17 votes are verified as valid. Figure 5.4 shows the average MOS values obtained from the subjective test and the calculated MOS values based on STVQM for the videos with different PSNR levels.

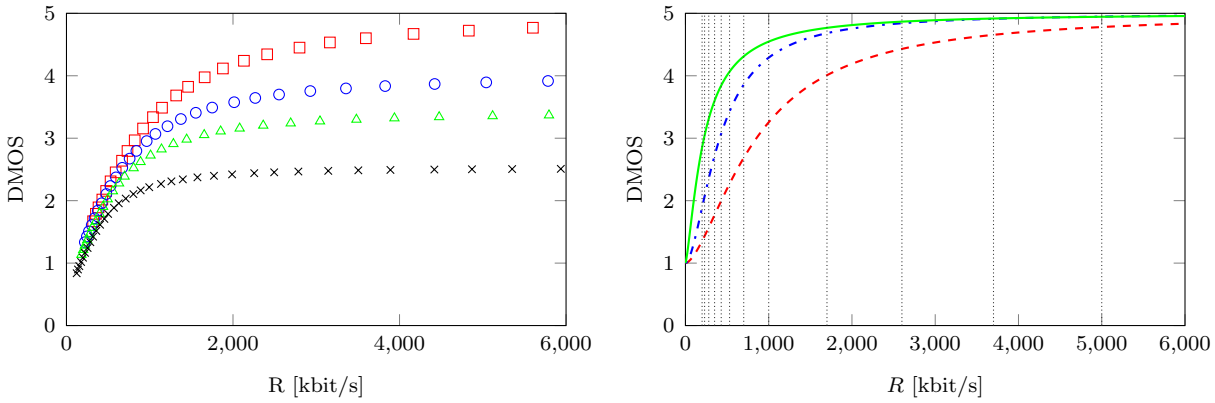
To create the rate-MOS curves, video sequences are created by encoding each SRC with a constant QP ($q \in \{10, \cdots, 50\}$) at different frame rates ($f \in \{5, 10, 15, 30\}$) with an IPPP... GoP structure. Different GoP lengths are used for each frame rate such that one GoP has a length of $\tau = 2\,\text{s}$, which is identical to the DASH segment length. For the encoded video sequences, MOS values are determined using the trained STVQM with the selected frame rates and the measured PSNR values. Figure 5.5a shows the determined R-D points for one of the selected videos. For each video, an exhaustive search is performed to select the encoding parameters q and f that maximize DMOS for the given rate constraints. In order to maximize the subjective quality for a target bit rate R_{target}, following optimization problem is considered:

$$\begin{aligned} (q_{\max}, f_{\max}) &= \arg\max_{q,f} Q(q,f) \\ &\text{subject to } R(q,f) \leq R_{\text{target}} \end{aligned} \tag{5.3}$$

where $Q(q,f)$ is the subjective quality in terms of DMOS and $R(q,f)$ is the video bit rate for a selected (q,f)-tuple. The solution of the optimization problem given in Eq. (5.3) yields the R-D points corresponding to the optimal encoding parameters $q_{\max}$ and $f_{\max}$. Next, rate-MOS curves are constructed using the determined R-D points. In [HE14], the authors propose to use a logistic function expressed in $\log(R, \text{MOS})$ to fit the R-D values. The logistic function is given as:

$$y = f(x) = a + \frac{b-a}{1 + exp[-c(x-d)]} \tag{5.4}$$

where the parameters a and b are set to the boundaries of the MOS rating scale, i.e., $a = 1$ and $b = 5$. Remaining parameters of the logistic function, c and d, are obtained through nonlinear curve-fitting that can be expressed in least-square sense [HE14]. The obtained rate-MOS curves can be used to create an arbitrary set of R-D points for each video. Hence, DMOS values corresponding to any bit rate can be calculated. Figure 5.5b shows the generated rate-MOS curves for the three considered videos.



(a) R-D points of Video 1 corresponding to different QP values $q \in \{10, \cdots, 50\}$ for frame rates: 30 fps (□), 15fps (○), 10fps (△), and 5fps (×).

(b) Rate-MOS curves of Video 1 (- - -), Video 2 (· - · -), and Video 3 (——); bit rates of the selected video levels (·····).

Figure 5.5: R-D points of Video 1 and Rate-MOS curves for all of the selected videos.

5.1.3 DASH video level selection

Selection of the set of video levels has a significant impact on the user satisfaction in DASH [TAPS+14]. In this section, the full video level set $\mathbf{V}$ for uplink streaming is constructed using the measured network performance characteristics (from Chapter 4) and the developed perceptual R-D model (from Section 5.1.2).

Most of the commercial AHS systems such as Microsoft Smooth Streaming [Zam09], Apple Live Streaming [PM10], and Adobe Dynamic streaming [Ado10] present recommendations for the selection of representation sets. Table 5.2 shows the video levels offered by some proprietary implementations for certain video resolutions. However, there is no scientific study in the literature that justifies the selection of those representation sets [TAPS+14]. Toni et. al. [TAPS+14] showed that the existing recommended sets of the proprietary implementations have critical weaknesses. They presented and solved an optimization problem to select the best encoding parameters for a representation set in an AHS system. Moreover, based on the solution of this optimization problem, they formulated generic

guidelines which are useful for the selection of the optimal representation set. In the present thesis, those guidelines are followed to select the video levels.

Table 5.2: Bit rate recommendations of industry solutions for multi-rate streaming (adapted from [GTH+13]).

Resolution	Bit rate [kbit/s]	Streaming solution
1920x1080	6000, 5000	Microsoft Smooth Streaming
	8000, 6000, 5500, 5000, 4000	Adobe HTTP Dynamic Streaming
	7000-8000	Apple QuickTime
	8000-50000 (upload)	YouTube
1280x720	3450, 2272, 1672	Adobe Flash Media Server
	4000, 3500, 3000, 2500, 2000, 1500	Adobe HTTP Dynamic Streaming
	4500, 2500, 1800	Apple HLS
	5000-6000	Apple QuickTime
	3450, 3000, 2100, 1400	Microsoft Smooth Streaming
	5000-30000 (upload); 2400 (live)	YouTube
	3500	MTV
640x480	1200, 600	Apple HLS
	1000-2000	Apple QuickTime

For the number of video levels M in $\mathbf{V}$, typical CDN settings are employed and M is set to 12. Two factors are considered to determine the distribution of video levels in $\mathbf{V}$. First, the lowest and highest video levels are determined such that the throughput range observed from the measurements in HSPA and LTE networks is covered. Secondly, more emphasis is put on lower bit rates ($R \leq 1000\,\text{kbit/s}$) as suggested by [TAPS+14]. The developed perceptual R-D model also shows that encoding a large number of video levels at lower rates is valuable since the gains in terms of subjective quality are usually large in this range (Figure 5.5b). This means that a small increase in throughput results in a significant subjective quality gain. Conversely, subjective quality increase is negligible at higher bit rates after a certain threshold. Figure 5.5b indicates that this threshold is 5000 kbit/s. Thus, the highest video level in $\mathbf{V}$ is set to 5000 kbit/s.

Taking these two factors and typical CDN settings into account, the full video level set $\mathbf{V}$ is constructed as follows (in kbit/s):

$$\mathbf{V} = \{200, 230, 280, 350, 430, 530, 700, 1000, 1700, 2600, 3700, 5000\}$$

The video levels in $\mathbf{V}$ are shown in Figure 5.5b as dotted vertical lines.

5.2 Performance comparison

A simulation framework was implemented in MATLAB to evaluate the performances of three client rate adaptation algorithms from Liu [LBG11], Miller [MQGW12] and Tian [TL13] introduced in Section 3.2. In this section, performances of these algorithms in terms of QoE are investigated. It is assumed that the video is streamed live, and the encoder generates each segment in multiple bit rates using the full video level set $\mathbf{V}$ defined in Section 5.1.3. For the performance analysis, four key performance indicators (KPIs) are introduced which are determined at the DASH client side. These KPIs are denoted by σ, ν, μ_r, and μ_m. σ is defined as the total duration of *stalling* events in a streaming session when the buffer is empty, and the playback of the video stream is interrupted. A high interrupted playback duration leads to a low user satisfaction [SES+14]. The number of video level switches (ν) in a session indicates the amount of quality changes in a streaming session. A lower number of switches during a session leads to a higher overall user experience [NEE+11b]. ν is calculated as:

$$\nu = \sum_{i=0}^{I} a(v_i), \qquad a(v_i) = \begin{cases} 0 & v_{i-1} = v_i \\ 1 & v_{i-1} \neq v_i,\ i = 0 \end{cases} \tag{5.5}$$

where i is the index of the requested segments, I is the number of fetched segments in the session and v_i the video rate of the i-th fetched segment. The *mean video rate* μ_r defines the mean bit rate of the delivered video levels for the streaming session and is determined by:

$$\mu_r = \frac{1}{I} \cdot \sum_{i=1}^{I} v_i, \tag{5.6}$$

Lastly, μ_m represents the mean MOS of the delivered video levels and is calculated as:

$$\mu_m = \frac{1}{I} \cdot \sum_{i=1}^{I} m(v_i), \tag{5.7}$$

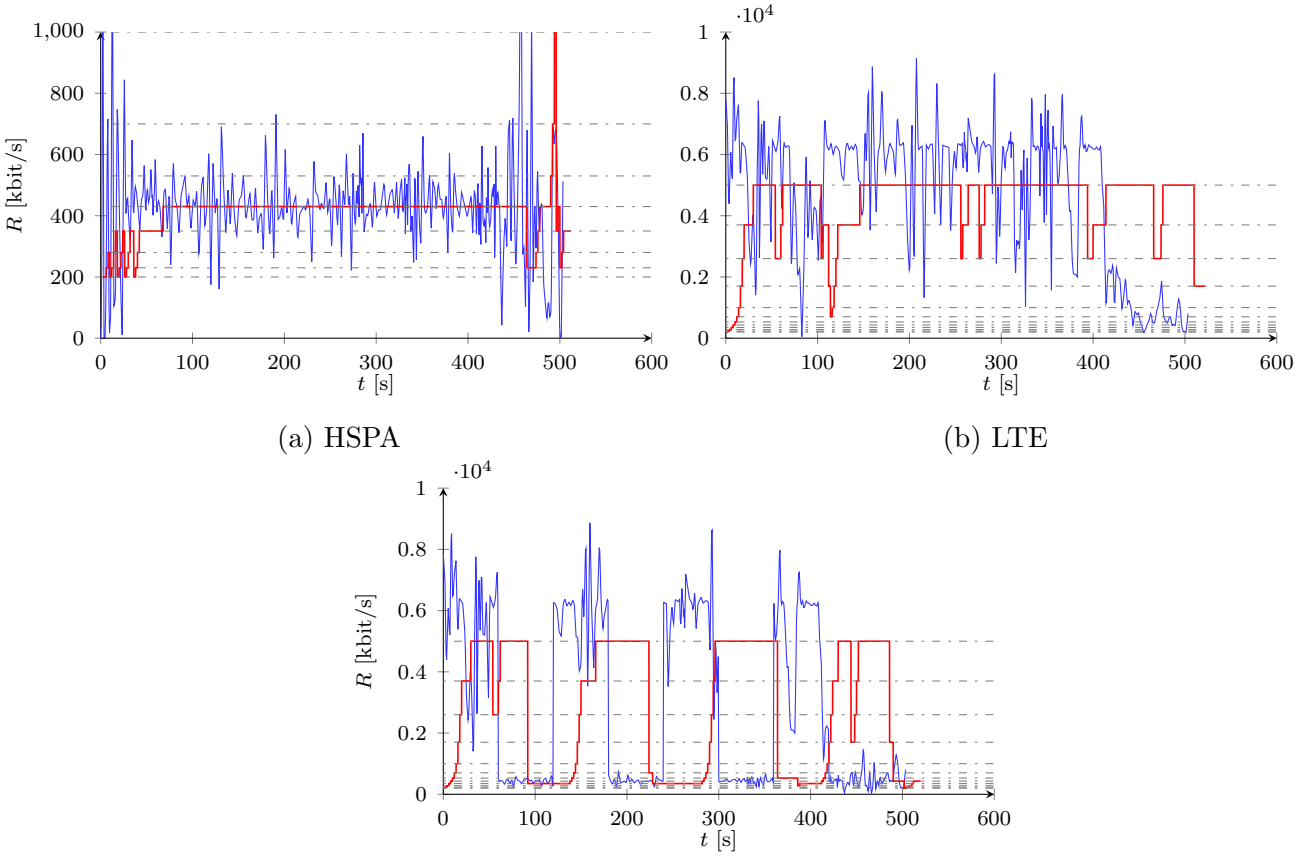
where $m(v_i)$ is the MOS value for v_i based on the perceptual R-D model developed in Section 5.1.2.

5.2.1 Liu's algorithm

Details of Liu's algorithm are explained in Section 3.2.1. The algorithm has only two parameters: switch-down threshold (γ) and switch-up threshold (ϵ). For performance evaluation, the switch-down threshold is set to $\gamma = 0.67$. This means that the algorithm

decides to switch down only when the ratio (μ) of segment duration (τ) to segment fetch time is smaller than 0.67 ($\mu < 0.67$). Otherwise, the algorithm either keeps the same video level or switches one level up. Switch-up occurs only if μ exceeds $(1 + \epsilon)$ where ϵ depends on the employed video level set and is calculated using Eq. (3.2). Using the determined video level set $\mathbf{V}$ from Section 5.1.3, the switch-up threshold is set to $\epsilon = 0.7$, thus the algorithm switches up only if $\mu > 1.7$. The algorithm also leaves an *idle time* after the download of the current segment to delay the download of the next segment if the buffer fullness exceeds a certain threshold. The idle time was arranged to obtain an average buffer fullness around 30 s.

Figure 5.6 shows the playout curve of the video levels selected by Liu's rate adaptation algorithm and the available uplink TCP throughput in the urban scenario for selected single RAN and handover traces.



(a) HSPA

(b) LTE

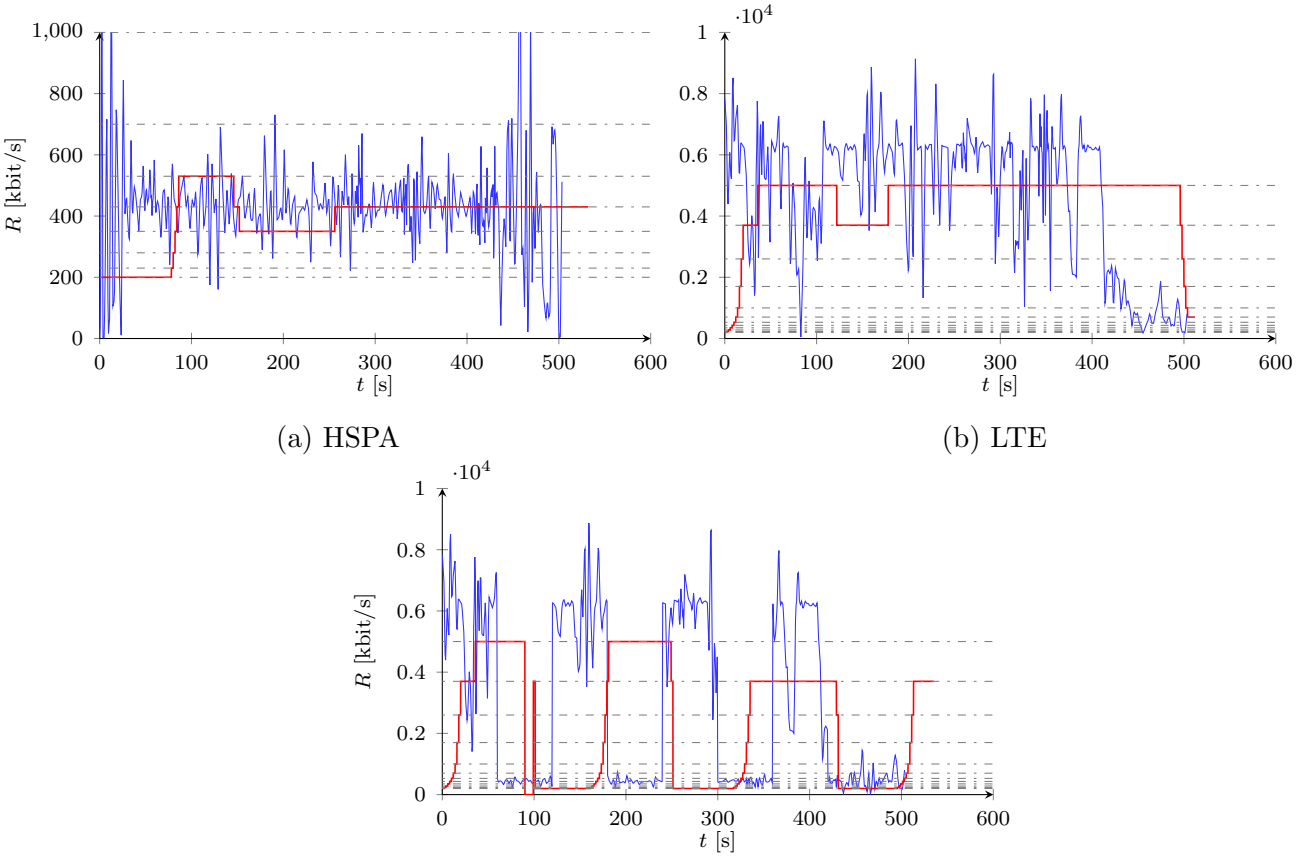
(c) Handover

Figure 5.6: Exemplary network throughput traces (——) in the urban scenario with the video level selections (——) using Liu's algorithm; video levels of $\mathbf{V}$ (- - -).

5.2.2 Miller's algorithm

Details of Miller's algorithm are explained in Section 3.2.2. The algorithm has a set of buffer-related parameters $\{B_{\text{min}}, B_{\text{low}}, B_{\text{high}}\}$ and a set of safety margins $\{\alpha_1, \alpha_2, \alpha_3, \alpha_4, \alpha_5\}$ which tune the sensitivity of the switch-up and switch-down decisions. The algorithm tries to keep the buffer level close to the middle of the target interval $\mathcal{B}_{\text{tar}} = [B_{\text{low}}, B_{\text{high}}]$. Therefore, the buffer-related parameters are set to: $B_{\text{min}} = 5\,\text{s}$, $B_{\text{low}} = 20\,\text{s}$, $B_{\text{high}} = 40\,\text{s}$ which offer a mean buffer level around 30 s. The safety margins are set to $\{0.75, 0.33, 0.5, 0.75, 0.9\}$ which are the default values suggested by Miller et al. [MQGW12].

Figure 5.7 shows the playout curve of the video levels selected by Miller's rate adaptation algorithm and the available uplink TCP throughput in the urban scenario for selected single RAN and handover traces.



(a) HSPA

(b) LTE

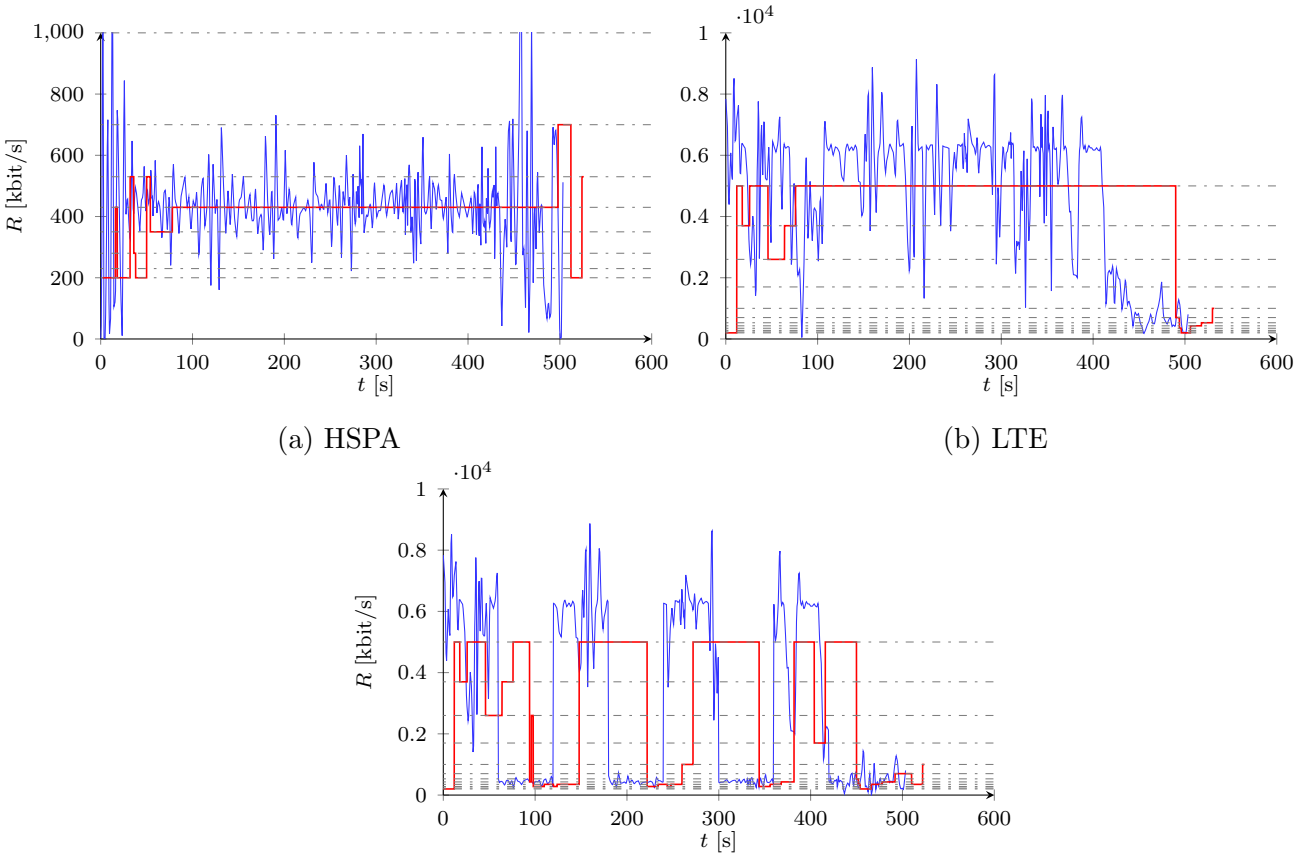
(c) Handover

Figure 5.7: Exemplary network throughput traces (——) in the urban scenario with the video level selections (——) using Miller's algorithm; video levels of $\mathbf{V}$ (· · · ·).

5.2.3 Tian's algorithm

Details of Tian's algorithm are explained in Section 3.2.3. The algorithm has two buffer-related parameters, q_{thr} and q_{cap}. It switches down if the buffer level falls below $q_{\text{thr}}/2$ and delays the download of the next segment if the buffer level exceeds q_{cap}. To ensure a mean buffer level around 30 s, these parameters are set to $q_{\text{thr}} = 40\,\text{s}$ and $q_{\text{cap}} = 40\,\text{s}$. The number of historical TCP measurements that is used for TCP throughput estimation is set to $K = 5$ segments, and the smoothing parameter is set to $m = 5$ (cf. Section 3.2.2).

Figure 5.8 shows the playout curve of the video levels selected by Tian's rate adaptation algorithm and the available uplink TCP throughput in the urban scenario for selected single RAN and handover traces.



(a) HSPA

(b) LTE

(c) Handover

Figure 5.8: Exemplary network throughput traces (——) in the urban scenario with the video level selections (——) using Tian's algorithm; video levels of $\mathbf{V}$ (- - -).

5.2.4 Evaluation

Figures 5.6 to 5.8 demonstrate that the selected video levels follow the TCP throughput fluctuations closely for Liu's, Miller's, and Tian's client-side rate adaptation algorithms. It is observed that the selected video levels remain stable in the HSPA scenarios due to relatively small throughput fluctuations. However, larger throughput fluctuations are observed in the LTE scenarios. Hence, large video quality shifts occur and more video levels are needed for rate adaptation during the streaming session. Similarly, large video quality shifts are observed in inter-RAN handover scenarios.

So far, performance of the rate adaptation algorithms has been investigated based on an individual uplink TCP throughput measurement. However, performance of the rate adaptation algorithms varies remarkably depending on the TCP throughput process, i.e., the network trace employed in the experiment. Therefore, multiple experiments have to be considered to evaluate the performance of the rate adaptation algorithms. To this end, the mean performance for all four KPIs ($\overline{\sigma}$, $\overline{\nu}$, $\overline{\mu_r}$, and $\overline{\mu_m}$) is investigated over all TCP uplink throughput traces of the single RAN (HSPA, LTE) and handover scenarios. Table 5.3 displays the results for all three client-side adaptation algorithms.

Table 5.3: Performance overview of the rate adaptation algorithms from Liu [LBG11], Miller [MQGW12] and Tian [TL13]; mean over nine traces each.

		$\overline{\sigma}$ [s]	$\overline{\nu}$	$\overline{\mu_r}$ [kbit/s]	$\overline{\mu_m}$ [MOS]		
					Video 1	Video 2	Video 3
HSPA	Liu	0.0	21	401.0	1.92	2.94	3.74
	Miller	0.0	4	337.5	1.74	2.64	3.53
	Tian	0.0	8	410.1	1.94	2.97	3.77
LTE	Liu	0.0	37	4365.2	4.58	4.84	4.88
	Miller	0.0	19	4321.9	4.54	4.80	4.85
	Tian	0.0	13	4383.6	4.47	4.73	4.81
Handover	Liu	0.0	49	2804.5	3.57	4.14	4.46
	Miller	4.9	47	2190.4	3.21	3.65	4.07
	Tian	0.8	26	2775.7	3.44	3.95	4.33

The results show that in single RAN scenarios, no interruption of playback due to buffer underruns is observed for all of the considered rate adaptation algorithms. In terms of the number of video quality switches, Miller's and Tian's algorithms outperform Liu's algorithm. However, Liu's algorithm provides a better mean video rate and mean MOS than Miller's algorithm. The difference is especially remarkable in handover scenarios. Mean MOS values obtained using Liu's and Tian's algorithms are comparable. Table 5.3 indicates that Tian's algorithm offers both high mean MOS and low number of video quality shifts in all scenarios. Therefore, Tian's algorithm provides the best perceptual quality among all three during the streaming process.

Another important point that Table 5.3 reveals is the content dependency of perceptual

quality. It is observed that the calculated mean MOS values for Video 1, Video 2, and Video 3 are very different. Moreover, the difference between the mean MOS values obtained with Video 1, which has the highest TA and SA values; and Video 3, which has the lowest TA and SA values, is greater when the streaming is performed using HSPA. For HSPA, the mean MOS difference between Video 1 and Video 3 is 1.81, whereas it is only 0.32 for LTE. These results can be explained by the sigmoid shape of the rate-MOS curves, cf. Figure 5.5b.

Chapter conclusion

In this chapter a vehicular DASH architecture is proposed to upstream videos from a vehicle's front-facing camera. A perceptual R-D model is developed to determine the DASH video levels in a perceptual quality-aware manner and take perceptual quality into account for performance evaluation. Using the proposed vehicular DASH architecture and the developed perceptual R-D model, performances of the standard DASH client adaptation algorithms from Liu et al. [LBG11], Miller et al. [MQGW12], and Tian et al. [TL12] are evaluated in single RAN (HSPA, LTE) and handover scenarios. Also, through exemplary network traces, distinct rate adaptation strategies of the different client adaptation algorithms are demonstrated.

Chapter 6

Dynamic video level encoding for uplink DASH

6.1 Objective and methodology

Rate adaptation algorithms presented in Chapter 5 allow the client to choose an appropriate video level from a static, full set of video levels according to metrics such as estimated available TCP throughput and buffer status. However, encoding a set of 10-15 video levels in parallel is not always feasible in a resource-constrained, vehicular streaming scenario since the ECUs in a vehicle might not be able to create the same number of video levels in real-time as CDN systems [LGSS]. One option to reduce the computational demand of the process of encoding the video levels would be to use SVC with DASH [MRL+12], but it is still unclear whether the scalable extensions of the codecs H.264/AVC and H.265/HEVC will find broad acceptance in the market [LGSS].

In this chapter, a live video delivery scenario is considered where a video is streamed from a vehicle to a video portal using DASH. Considering this scenario, different encoder side video level selection approaches are proposed which reduce the number of video levels that need to be encoded. To this end, three algorithms are developed which select a reduced set of video levels from a static, pre-defined set based on either TCP uplink throughput information or client request history. The first two algorithms are *network-aware*, i.e., they employ the measured or historical TCP uplink throughput information to reduce the number of the encoded video levels. The first algorithm uses only the geo-referenced network statistics obtained from a remote bandwidth lookup database whereas the second algorithm additionally uses the measured network performance of the source. The third algorithm is fundamentally different from the first two since it considers the video level request history of the client instead of the network context information.

In the following, the proposed system models are introduced, the network-aware and client

request-based video level selection algorithms are described (Section 6.2 - 6.3), and the performances of the respective video level selection algorithms are evaluated (Section 6.4).

6.2 Network-aware video level selection

6.2.1 System model

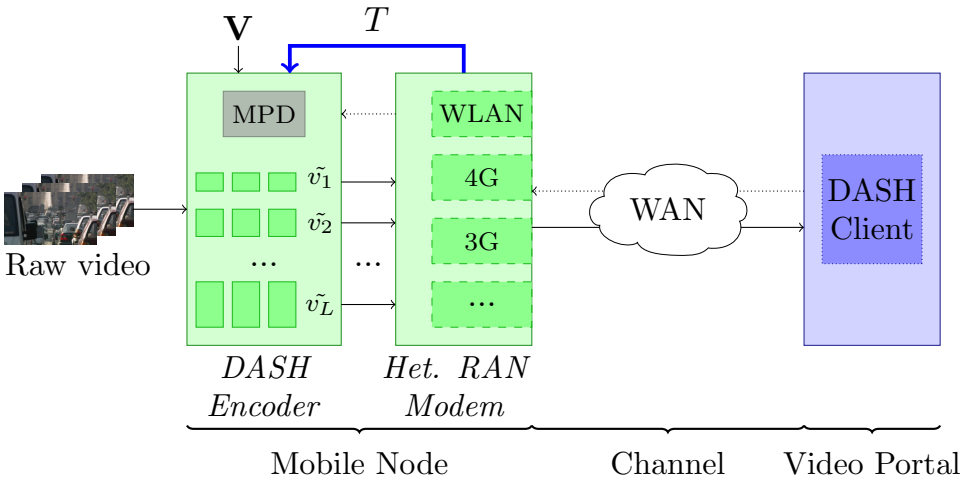


Figure 6.1: System model for the TCP throughput-based adaptation of the video levels at the encoder side [LGSS].

Figure 6.1 displays the considered system model for the network-aware encoder side video level selection. A DASH-enabled encoder is set up in one of the ECUs of the vehicle. The encoder splits the video into segments of length τ, encodes the segments at different bit rates, and stores them in a locally installed web server. All segments are referenced in an MPD file which informs the client about the available bit rates. The full set of possible representations is denoted by a static set $\mathbf{V}$ which consists of M video levels at pre-defined bit rates. A DASH client is installed on a video portal which adaptively requests segments from the vehicle using a standard DASH adaptation algorithm, e.g., one of the algorithms described in Section 3.2. The end-to-end network architecture between the vehicle and video portal is assumed to be as described in Section 3.4.1.

It is assumed that the heterogeneous RAN modem of the vehicle provides information about the measured uplink TCP throughput (T_{Meas}) for a target window length of N seconds. At the session start-up, no TCP throughput measurements are available yet. Also, after an inter-RAN handover the existing TCP throughput history is no longer reliable due to large shifts of the available TCP throughput during an inter-RAN handover. Hence, it is assumed that a remote, geo-referenced bandwidth lookup database exists which provides information

about the statistics of the TCP uplink throughput (T_{DB}) for the current position and the connected RAN of the vehicle [LGSS].

The remote database is created through previous throughput measurements of other vehicles on the same route. In a typical streaming session, vehicles request statistics of the available TCP throughput from the remote database, and they are also responsible for sending their own location and their measured TCP throughput information. Vehicles collect the required data and then upload them to a remote database which builds a bandwidth map and correlates location information with the measured TCP throughput.

It is assumed that the standard DASH client adaptation algorithm at the video portal (i) downloads the video segments in chronological order, (ii) downloads each segment at exactly one video level, and (iii) downloads the segments in a non-preemptive way, i.e., the download of segment $i-1$ must be completed before the start of the download of the segment i.

6.2.2 Cooperative adaptation algorithm

The goal of the algorithm is to dynamically select a reduced set of video levels $\tilde{\mathbf{V}}$ from a static set $\mathbf{V}$ based on the statistics of the TCP uplink throughput information received from a remote database.

The algorithm's pseudo-code is provided in Algorithm 4. It is assumed that the algorithm is invoked at time t immediately after the download of a segment when one of the two following cases occurs: (i) a new streaming session is started or an inter-RAN handover between RANs occurs, (ii) a window of N seconds is completed. It is necessary to re-invoke the algorithm after an inter-RAN handover since otherwise, the throughput differences between two RANs could mislead the algorithm to choose inappropriate video levels and degrade the performance.

The algorithm takes the following input arguments: (i) $\mathbf{V} = \{v_1, v_2, \cdots, v_M\}$: the static full set of pre-defined video levels, (ii) L: number of levels in the reduced set, (iii) T_{DB}: TCP uplink performance information obtained from a remote database. The algorithm has one output argument, $\tilde{\mathbf{V}} = \{\tilde{v}_1, \tilde{v}_2, \cdots, \tilde{v}_L\}$: the reduced set of video levels. The algorithm requires $L \leq M$ so that the number of encoded video levels is reduced.

Once the algorithm is invoked, the estimated available TCP uplink throughput T is set to the TCP uplink throughput T_{DB} requested from the remote database. To this end, the vehicle sends its current position and connected RAN to the remote database. In return, it receives the corresponding average TCP throughput T_{DB} over a target window of the next N seconds.

In the next step, the algorithm determines the video levels of $\tilde{\mathbf{V}}$ according to T. First, the video level v_n, which has the smallest absolute difference to T among the video levels in the full set $\mathbf{V}$, is selected as the center element of $\tilde{\mathbf{V}}$. The algorithm selects the $L-1$

remaining levels around v_n. If n is an odd number, the selection is symmetric. Otherwise, the algorithm takes a conservative approach and favors the video levels with smaller rates than v_n. Depending on L, if there are not enough video levels above (below) v_n, the algorithm constructs $\tilde{\mathbf{V}}$ asymmetrically around v_n. In this case, the algorithm selects all the available video levels above (below) v_n and the rest from below (above). This operation corresponds to selecting the last (first) L levels from $\mathbf{V}$.

Algorithm 4: COOPERATIVE ADAPTATION ALGORITHM

Input: $\mathbf{V} = \{v_1, v_2, \cdots, v_M\}$: full set of available video levels
L: number of levels in the reduced set
T_{DB}: TCP uplink throughput from remote database
Output: $\tilde{\mathbf{V}} = \{\tilde{v}_1, \tilde{v}_2, \cdots, \tilde{v}_L\}$: reduced set of video levels

$T = T_{\mathrm{DB}}$
$v_n = \underset{\{v_i, 1 \le i \le M\}}{\arg\min} \|v_i - T\|$
if $n < \lfloor {}^{L}/_{2} \rfloor + 1$ **then**
  $\tilde{\mathbf{V}} \leftarrow \{v_1, v_2, \cdots, v_L\}$
else if $n > M - \lceil {}^{L}/_{2} - 1 \rceil$ **then**
  $\tilde{\mathbf{V}} \leftarrow \{v_{M-L+1}, v_{M-L+2}, \cdots, v_M\}$
else
  if *L is odd* **then**
    $\tilde{\mathbf{V}} \leftarrow \{v_{n-\lfloor {}^{L}/_{2} \rfloor}, v_{n-\lfloor {}^{L}/_{2} \rfloor+1}, \cdots, v_{n+\lfloor {}^{L}/_{2} \rfloor}\}$
  else
    $\tilde{\mathbf{V}} \leftarrow \{v_{n-\frac{L}{2}}, v_{n-\frac{L}{2}+1}, \cdots, v_{n+\frac{L}{2}-1}\}$
return $\tilde{\mathbf{V}}$

6.2.3 History-based adaptation algorithm

The goal of the algorithm is to dynamically select a reduced set of video levels $\tilde{\mathbf{V}}$ from a static set $\mathbf{V}$ based on *both* the statistics of the TCP uplink throughput information received from a remote database (T_{DB}) and the measured network performance of the vehicle (T_{Meas}). The main motivation for this approach is to reduce the signalling overhead compared to the *Cooperative* implementation. This approach is also useful when T_{DB} deviates significantly from the actual TCP uplink throughput. In this case, the *Cooperative* algorithm could select inappropriate video levels which might lead to playout interruptions and perceptual quality degradation. Therefore, T_{Meas} might provide a more accurate throughput estimation. The algorithm's pseudo-code is provided in Algorithm 5.

In addition to the input arguments stated in the *Cooperative* algorithm, the algorithm takes two additional input arguments: (i) h: flag which is set to 1 after start-up and a inter-RAN handover, and 0 otherwise, (ii) T_{Meas}: measured TCP uplink throughput.

Once the algorithm is invoked, the TCP uplink throughput information is determined. After the start-up and after an inter-RAN handover, h is set to 1 and thus the algorithm uses the statistics of the TCP uplink throughput from a remote database as in the *Cooperative* algorithm. Thus, T is set to the TCP uplink throughput T_{DB} of the current position and connected RAN of the vehicle requested from the remote database. At all other times, h is set to 0. Hence, the algorithm sets T to T_{Meas}.

In the next step, the algorithm determines the video levels of $\tilde{\mathbf{V}}$ according to T. First, the video level v_n, which has the smallest absolute difference to T among the video levels in $\mathbf{V}$, is selected as the center element of $\tilde{\mathbf{V}}$ (line 5). The algorithm then determines the remaining $L-1$ video levels in the same way as the *Cooperative* algorithm (lines 6-14).

Algorithm 5: HISTORY-BASED ADAPTATION ALGORITHM

Input: $\mathbf{V} = \{v_1, v_2, \cdots, v_M\}$: full set of available video levels
L: number of levels in the reduced set
h: flag which is set to 1 at startup and handover
T_{Meas}: measured TCP uplink throughput
T_{DB}: TCP uplink throughput from remote database
Output: $\tilde{\mathbf{V}} = \{\tilde{v}_1, \tilde{v}_2, \cdots, \tilde{v}_L\}$: reduced set of video levels

1 **if** $h = 1$ **then**
2 $\quad T = T_{\mathrm{DB}}$
3 **else**
4 $\quad T = T_{\mathrm{Meas}}$
5 $v_n = \underset{\{v_i, 1 \le i \le M\}}{\arg\min} \; \|v_i - T\|$
6 **if** $n < \lfloor L/2 \rfloor + 1$ **then**
7 $\quad \tilde{\mathbf{V}} \leftarrow \{v_1, v_2, \cdots, v_L\}$
8 **else if** $n > M - \lceil L/2 - 1 \rceil$ **then**
9 $\quad \tilde{\mathbf{V}} \leftarrow \{v_{M-L+1}, v_{M-L+2}, \cdots, v_M\}$
10 **else**
11 $\quad$ **if** *L is odd* **then**
12 $\quad\quad \tilde{\mathbf{V}} \leftarrow \{v_{n-\lfloor L/2 \rfloor}, v_{n-\lfloor L/2 \rfloor+1}, \cdots, v_{n+\lfloor L/2 \rfloor}\}$
13 $\quad$ **else**
14 $\quad\quad \tilde{\mathbf{V}} \leftarrow \{v_{n-\frac{L}{2}}, v_{n-\frac{L}{2}+1}, \cdots, v_{n+\frac{L}{2}-1}\}$
15 **return** $\tilde{\mathbf{V}}$

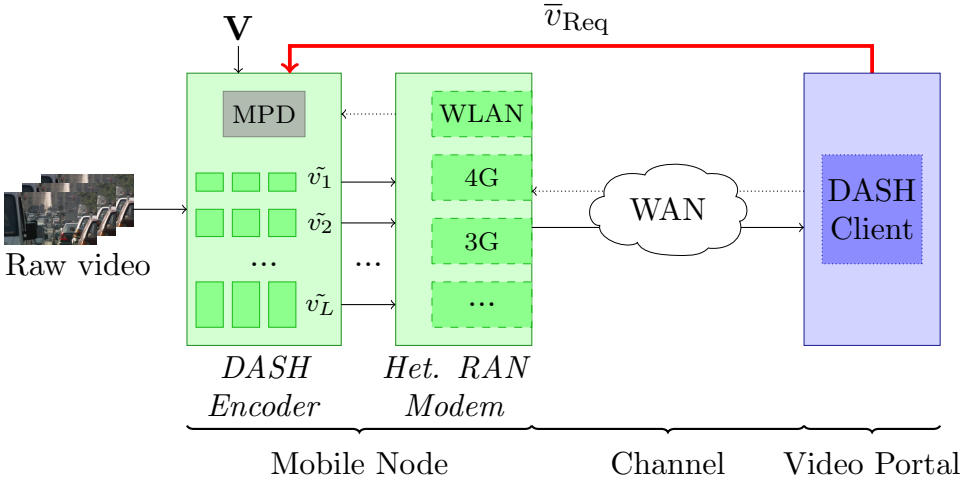


Figure 6.2: System model for the client request history-based adaptation of the video levels (adapted from [LGSS]).

6.3 Client request-based video level selection

6.3.1 System model

Figure 6.1 displays the considered system model for client request history-based video level selection at the encoder side. Different from the network-aware video level selection, the present implementation assumes that no TCP throughput information is employed in the process of dynamically encoding the video levels. Instead, the DASH encoder selects a reduced set of video levels based on the *request history* of the client. Here, the notion request history refers to the video levels that the client *would* request, if the full set of video levels $\mathbf{V}$ *were* available at each adaptation decision. The DASH server can keep track of both the actual HTTP requests of the DASH client based on a dynamically generated, reduced video level set $\tilde{\mathbf{V}}$ and the hypothetical HTTP requests based on $\mathbf{V}$. Next, the server calculates the average bit rate of the (hypothetically) requested video levels $\overline{v}_{\text{Req}}$ over a target window of N seconds. The server then uses $\overline{v}_{\text{Req}}$ to encode the new video levels and update $\tilde{\mathbf{V}}$.

El Essaili et al. [ESS+13] proposed a proxy-based approach to redirect the client HTTP requests to the closest lower representation from the MPD which matches the QoE optimization result. The present implementation is similar to their approach. However, the rewriting of the client HTTP requests is done at the server-side instead of a proxy and is based on the previous HTTP requests rather than a QoE optimization.

6.3.2 Client request-based adaptation algorithm

The goal of the algorithm is to dynamically select a reduced set of video levels $\tilde{\mathbf{V}}$ from the static set $\mathbf{V}$ based on the video level request history of the DASH client.

The algorithm's pseudocode is provided in Algorithm 6. It is assumed that the algorithm is invoked at time t immediately after the download of a segment when a window of N seconds is completed. Different from the network-aware video level selection algorithms, this algorithm does not distinguish between single RAN and handover scenarios.

The algorithm takes two input arguments: (i) $\mathbf{V} = \{v_1, v_2, \cdots, v_M\}$: the static full set of pre-defined video levels, (ii) L: number of levels in the reduced set. The algorithm has one output argument, $\tilde{\mathbf{V}} = \{\tilde{v}_1, \tilde{v}_2, \cdots, \tilde{v}_L\}$: the reduced set of video levels.

At the beginning of the streaming session, the MPD file with the full set of video levels ($\mathbf{V}$) is transmitted to the client. Since the encoder has not created a reduced set of video levels yet, the client selects the video levels from $\mathbf{V}$ for the first N seconds. After N seconds, the server calculates $\overline{v}_{\text{Req}}$ over the first N seconds of the streaming session by using the previous client HTTP requests.

In the next step, the algorithm determines the video levels of the reduced set $\tilde{\mathbf{V}}$ according to $\overline{v}_{\text{Req}}$. First, the video level v_n, which has the smallest absolute difference to $\overline{v}_{\text{Req}}$ among the video levels in the full set $\mathbf{V}$, is selected as the center element of $\tilde{\mathbf{V}}$. The remaining $L-1$ levels are selected around v_n in the same way as described for the *Cooperative* and *History-based* algorithms (lines 2-10).

Algorithm 6: CLIENT REQUEST-BASED ADAPTATION ALGORITHM

Input: $\mathbf{V} = \{v_1, v_2, \cdots, v_M\}$: full set of available video levels
L: number of levels in the reduced set
Output: $\tilde{\mathbf{V}} = \{\tilde{v}_1, \tilde{v}_2, \cdots, \tilde{v}_L\}$: reduced set of video levels

1 $v_n = \underset{\{v_i, 1 \le i \le M\}}{\arg\min} \|v_i - \overline{v}_{\text{Req}}\|$
2 **if** $n < \lfloor L/2 \rfloor + 1$ **then**
3 $\quad \tilde{\mathbf{V}} \leftarrow \{v_1, v_2, \cdots, v_L\}$
4 **else if** $n > M - \lceil L/2 - 1 \rceil$ **then**
5 $\quad \tilde{\mathbf{V}} \leftarrow \{v_{M-L+1}, v_{M-L+2}, \cdots, v_M\}$
6 **else**
7 $\quad$ **if** *L is odd* **then**
8 $\quad\quad \tilde{\mathbf{V}} \leftarrow \{v_{n-\lfloor L/2 \rfloor}, v_{n-\lfloor L/2 \rfloor+1}, \cdots, v_{n+\lfloor L/2 \rfloor}\}$
9 $\quad$ **else**
10 $\quad\quad \tilde{\mathbf{V}} \leftarrow \{v_{n-\frac{L}{2}}, v_{n-\frac{L}{2}+1}, \cdots, v_{n+\frac{L}{2}-1}\}$
11 **return** $\tilde{\mathbf{V}}$

6.4 Simulation results

For the performance analysis of the network-aware and client request-based video level selection algorithms, a simulation framework was implemented in MATLAB. Two different implementations were considered at the encoder side: (i) a reference implementation (referred to as *Reference*) where the DASH encoder produces the full set of M video levels without any encoder side video level selection, (ii) an implementation where either one of the network-aware approaches or the client request-based approach is applied. On the DASH client side, three different standard AHS adaptation algorithms from Liu et al. [LBG11], Miller et al. [MQGW12] and Tian et al. [TL13] were considered. The parameters of the client rate adaptation algorithms were set to offer a mean buffer fullness of 30 s in the LTE scenario, which is feasible for most live streaming scenarios [LGSS]. For the performance analysis, the four KPIs introduced in Section 5.2 (σ, ν, μ_r, and μ_m) are employed.

The two algorithm parameters encoder side adaptation time (N) and the number of video levels in the reduced set (L) are determined through a pre-study. First, the influence of N on the *History-based* implementation is investigated and the minimum number of video levels $L_{\min}$ is determined that is required to achieve the same user QoE as for the *Reference* implementation. Hence, for each $N \in \{10, 20, 40, 60\}$ s, the corresponding $L_{\min}$ is determined such that the user QoE, i.e. the mean subjective quality of the transmitted video segments (μ_m), interrupted playback time due to stalling events (σ), and the number of video level switches (ν) is equal to or better than the *Reference* implementation. This investigation is performed by using the TCP uplink throughput traces of LTE measurements described in Chapter 4 and Liu's algorithm [LBG11] is employed at the client side. Thus, influence of the client side adaptation dynamics is kept to a minimum since Liu's algorithm is the least sophisticated among the considered client adaptation algorithms. Table 6.1 displays the obtained $(N, L_{\min})$-tuples.

The results of the pre-study show that $L_{\min}$ increases for larger N values since the responsiveness of the *History-based* implementation to the throughput fluctuations decreases as N increases. The lowest possible $L_{\min}$ value ($L_{\min} = 2$) is achieved for $N = 10$ s. This is an acceptable number of video levels since most modern vehicles equipped with cameras can encode two video levels in parallel [LGSS].

Table 6.1: Feasible (N, L_{min})-pairs for Algorithm 5.

N	10	20	40	60
L_{min}	2	4	4	5

Hence, the performance of the history-based video level selection algorithm is evaluated with the chosen parameters ($N = 10, L = 2$). The mean performance for all four KPIs is calculated over all TCP uplink throughput traces of the urban single RAN (LTE) and urban

handover scenarios. To provide a fair comparison, performances of the other proposed network-aware algorithm (*Cooperative*) is also evaluated using the same (N, L)-tuple.

For the client request-based algorithm, an initial investigation revealed that there is a significant performance degradation when less than four video levels are used. Hence, for a similar QoE as the *Reference* implementation, the parameters of the *Request-based* algorithm are set to $(N = 10, L = 4)$.

6.4.1 Cooperative algorithm

The results for the *Cooperative* algorithm are displayed in Table 6.2 for the three client side adaptation algorithms from Liu et al. [LBG11], Miller et al. [MQGW12], and Tian et al. [TL13]. It is observed that the performance of *Cooperative* implementation in terms of mean subjective quality and interrupted playback time due to stalling events is comparable to the performance of the *Reference* implementation for all three client side adaptation algorithms. Moreover, for Liu's [LBG11] and Miller's [MQGW12] algorithms, a decrease in the number of quality switches is observed, both for LTE and handover scenarios. The main reason for this effect is the small number of encoded video levels ($L = 2$), which decrease the probability of a quality switch.

Furthermore, it is observed that the mean video rate obtained with the *Cooperative* implementation is mostly lower than the mean video rate obtained with the *Reference* implementation (except for the handover scenario when Liu's algorithm is used). However, this does not necessarily lead to a lower mean subjective quality of the transmitted video segments over the streaming session due to the sigmoid shape of the rate-MOS curves (cf. Figure 5.5b). This implies that the mean subjective quality depends highly on which particular video levels are transmitted during a streaming session. This is best observed in the handover scenario when Miller's algorithm is used. The mean video rate of the *Cooperative* implementation is roughly 300 kbit/s lower than that of the *Reference* implementation. However, the mean subjective quality of the *Cooperative* implementation for Video 2 and Video 3 is higher, despite its lower mean video rate.

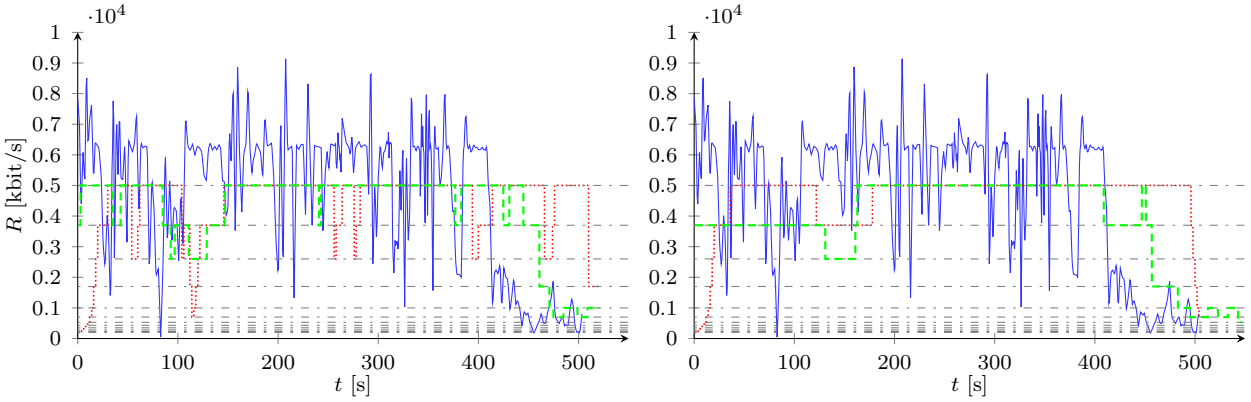
Figure 6.3 shows the transmitted video segments for an exemplary video streaming scenario using one LTE network trace for the *Cooperative* and *Reference* implementations, respectively. It is observed that the *Cooperative* algorithm requests higher video levels at the start-up phase since the algorithm already adapts the video level set at $t = 0$ based on the network statistics T_{DB} whereas the *Reference* implementation always starts with the lowest video level in $\mathbf{V}$ since it does not have any a priori information about the future available TCP throughput.

6.4.2 History-based algorithm

The results for the *History-based* algorithm are displayed in Table 6.3 for the three client

Table 6.2: Performance overview of the *Reference* and *Cooperative* implementations in urban and handover scenarios for $L = 2$, $N = 10$ s, mean over nine traces each.

		$\overline{\sigma}$ [s]		$\overline{\nu}$		$\overline{\mu_r}$ [kbit/s]		$\overline{\mu_m}$ [MOS] Video 1		Video 2		Video 3	
		Ref.	*Coop.*	*Ref.*	*Coop.*	*Ref.*	*Coop.*	*Ref.*	*Coop.*	*Ref.*	*Coop.*	*Ref.*	*Coop.*
LTE	Liu	0.0	2.3	37	22	4365.2	4278.5	4.58	4.56	4.84	4.85	4.88	4.88
	Miller	0.0	0.0	19	12	4321.9	3755.2	4.54	4.46	4.80	4.81	4.85	4.86
	Tian	0.0	0.2	13	19	4383.6	4044.0	4.47	4.52	4.73	4.83	4.81	4.87
Handover	Liu	0.0	1.8	49	23	2804.5	2858.2	3.57	3.55	4.14	4.13	4.46	4.47
	Miller	4.9	6.6	47	13	2190.4	1886.3	3.21	3.18	3.65	3.88	4.07	4.31
	Tian	0.8	1.4	26	22	2775.7	2438.4	3.44	3.35	3.95	3.98	4.33	4.37



(a) Liu [LBG11]

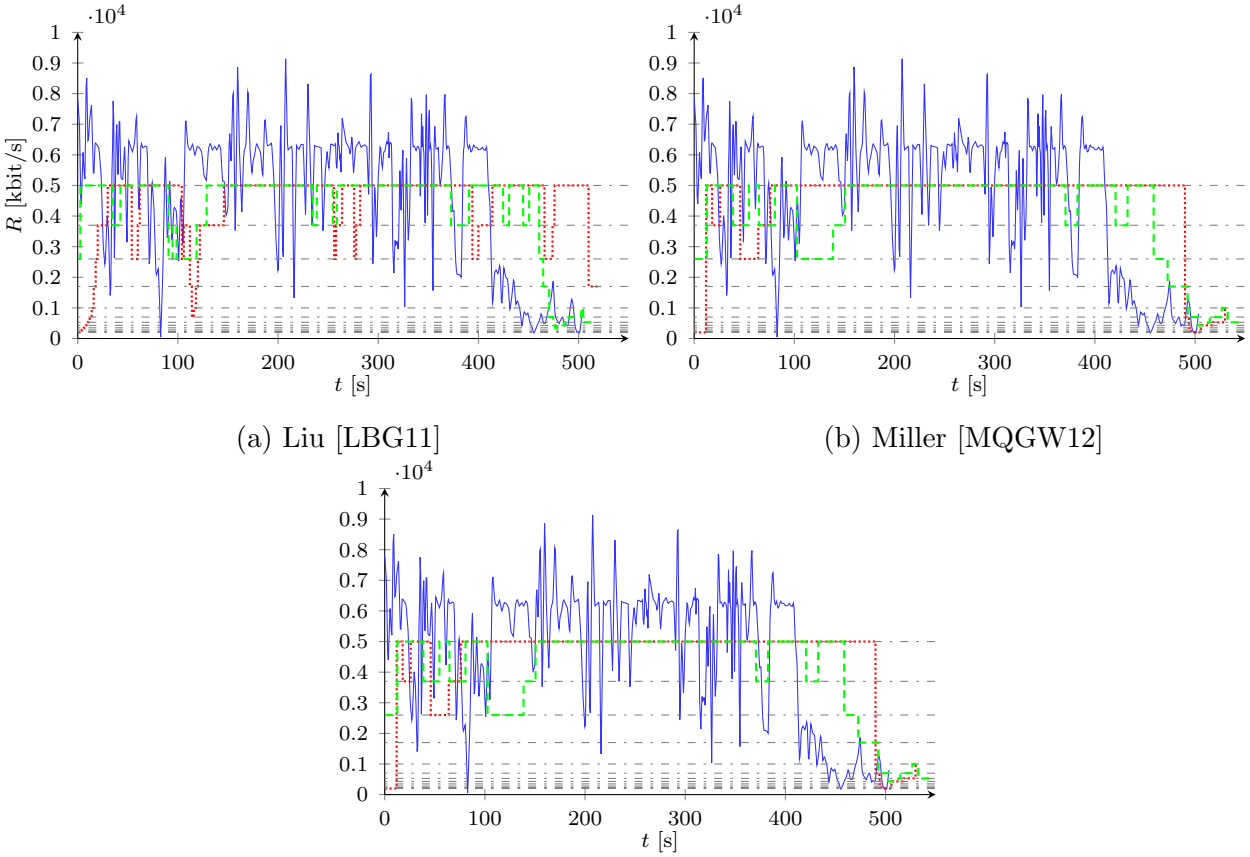
(b) Miller [MQGW12]

(c) Tian [TL13]

Figure 6.3: Exemplary LTE TCP uplink network throughput trace (——) with the video level decisions for the *Reference* (·····) and *Cooperative* (- - - -) implementations for different client adaptation algorithms ($N = 10, L = 2$); video levels of $\mathbf{V}$ (· · · ·).

Table 6.3: Performance overview of the *Reference* and *History-based* implementations in urban and handover scenarios for $L = 2$, $N = 10\,\mathrm{s}$, mean over nine traces each.

		$\overline{\sigma}$ [s]		$\overline{\nu}$		$\overline{\mu_r}$ [kbit/s]		$\overline{\mu_m}$ [MOS] Video 1		Video 2		Video 3	
		Ref.	*Hist.*	*Ref.*	*Hist.*	*Ref.*	*Hist.*	*Ref.*	*Hist.*	*Ref.*	*Hist.*	*Ref.*	*Hist.*
LTE	Liu	0.0	0.0	37	27	4365.2	4203.5	4.58	4.52	4.84	4.81	4.88	4.86
	Miller	0.0	0.0	19	16	4321.9	3544.0	4.54	4.41	4.80	4.77	4.85	4.84
	Tian	0.0	0.0	13	21	4383.6	3932.3	4.47	4.45	4.73	4.78	4.81	4.85
Handover	Liu	0.0	6.4	49	22	2804.5	2912.7	3.57	3.62	4.14	4.17	4.46	4.48
	Miller	4.9	6.2	47	20	2190.4	1928.8	3.21	3.27	3.65	3.95	4.07	4.35
	Tian	0.8	0.4	26	25	2775.7	2632.7	3.44	3.54	3.95	4.11	4.33	4.45



(a) Liu [LBG11]

(b) Miller [MQGW12]

(c) Tian [TL13]

Figure 6.4: Exemplary LTE TCP uplink network throughput trace (——) with the video level decisions for the *Reference* (·····) and *History-based* (- - - -) implementations for different client adaptation algorithms ($N = 10, L = 2$); video levels of $\mathbf{V}$ (- · - · -) [LGSS].

side adaptation algorithms from Liu et al. [LBG11], Miller et al. [MQGW12], and Tian et al. [TL13]. As expected from the pre-study, the performance of the *History-based* implementation in terms of mean subjective quality and interrupted playback time due to stalling events is comparable to the performance of the *Reference* implementation for all three client side adaptation algorithms.

Similar to the *Cooperative* implementation, a decrease in the number of quality switches observed when Liu's [LBG11] and Miller's [MQGW12] algorithms are used. However, the number of quality switches is in most cases slightly higher than that of the *Cooperative* implementation. The interrupted playback time, mean video rate, and mean subjective quality are around the same level as for the *Cooperative* implementation. However, it is observed that the selected video levels during the streaming process can be significantly different. Figure 6.4 displays the video level selections of the *History-based* implementation over an exemplary LTE network trace.

6.4.3 Client request-based algorithm

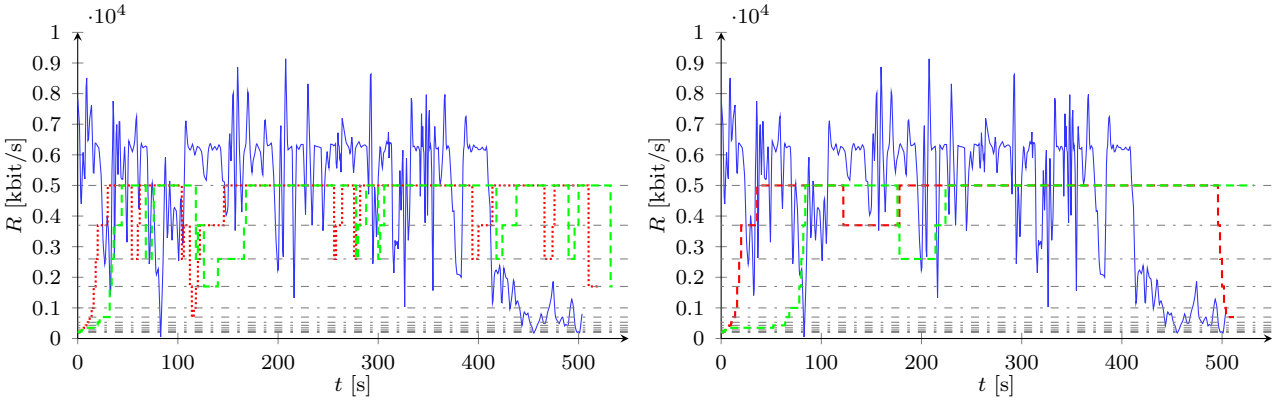
The results for the *Request-based* algorithm are displayed in Table 6.4 for the three client side adaptation algorithms from Liu et al. [LBG11], Miller et al. [MQGW12], and Tian et al. [TL13]. For the selected parameters $N = 10, L = 4$, performance of the *Request-based* implementation in terms of mean subjective quality is comparable to the performance of the *Reference* implementation for all three client side adaptation algorithms.

However, interrupted playback time due to stalling events is significantly higher in handover scenarios, especially when Miller's algorithm is used. The main reason for this effect is the indifference of the *Request-based* algorithm to different RANs. Since the video level selection at the server side is only a function of the request history of the DASH client (and not of the available TCP throughput), the selected video level set might still consist of inappropriately high video levels after, for example, a handover from LTE to HSPA occurs. Hence, more buffer underflows might occur which lead to more frequent and longer playback interruptions.

Another drawback of this approach is the low video level requests of the client at the start-up (compared to *History-based* and *Cooperative* implementations) since a pre-selection of video levels cannot be realized before the beginning of the streaming session. This is caused by the fact that no TCP network performance information is employed in the *Request-based* implementation. Therefore, the client, by default, requests the lowest video level in the full set $\mathbf{V}$ at the start-up. Figure 6.5 displays the video level selections of the *Request-based* implementation over an exemplary LTE network trace

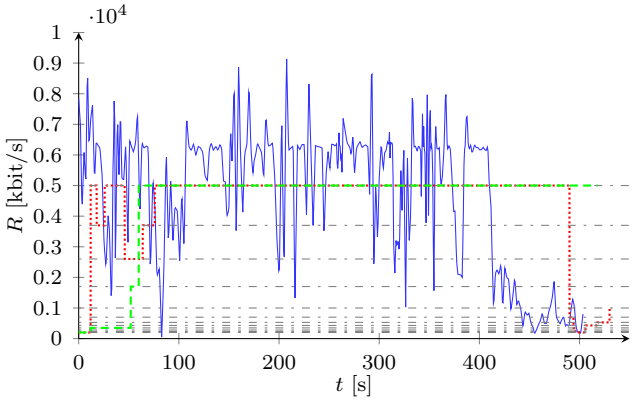
Table 6.4: Performance overview of the *Reference* and *Request-based* implementations in urban and handover scenarios for $L = 4$, $N = 10\,\mathrm{s}$, mean over nine traces each.

		$\overline{\sigma}$ [s]		$\overline{\nu}$		$\overline{\mu_r}$ [kbit/s]		$\overline{\mu_m}$ [MOS] Video 1		Video 2		Video 3	
		Ref.	*Req.*	*Ref.*	*Req.*	*Ref.*	*Req.*	*Ref.*	*Req.*	*Ref.*	*Req.*	*Ref.*	*Req.*
LTE	Liu	0.0	0.0	37	33	4365.2	4191.8	4.58	4.50	4.84	4.78	4.88	4.84
	Miller	0.0	0.0	19	15	4321.9	4062.2	4.54	4.29	4.80	4.62	4.85	4.74
	Tian	0.0	0.0	13	5	4383.6	4364.9	4.47	4.43	4.73	4.69	4.81	4.78
Handover	Liu	0.0	1.7	49	44	2804.5	2408.3	3.57	3.34	4.14	3.95	4.46	4.34
	Miller	4.9	17.3	47	35	2190.4	2295.7	3.21	3.11	3.65	3.61	4.07	4.07
	Tian	0.8	5.0	26	26	2775.7	2720.0	3.44	3.45	3.95	4.01	4.33	4.38



(a) Liu [LBG11]

(b) Miller [MQGW12]



(c) Tian [TL13]

Figure 6.5: Exemplary LTE TCP uplink network throughput trace (——) with the video level decisions for the *Reference* (·····) and *Client request-based* (- - - -) implementations for different client adaptation algorithms ($N = 10, L = 4$); video levels of $\mathbf{V}$ (· · · ·).

Chapter conclusion

Simulation results for all three encoder side video level selection algorithms show that the number of video levels that need to be encoded can significantly be reduced while maintaining a similar QoE as the reference implementation which uses a full, static set of video levels. Specifically, for an encoder side adaptation time of $N = 10\,\mathrm{s}$, the network-aware selection algorithms (*Cooperative* and *History-based*) achieve a reduction of 83% (from 12 down to 2 video levels) whereas the client request-based selection algorithm (*Request-based*) achieves a reduction of 67% (from 12 down to 4 levels).

Chapter 7

Conclusions and further research

7.1 Conclusions

In conclusion, the methodology and the main results of this work is summarized.

In this thesis, first, repeated TCP throughput measurements have been conducted in urban and highway vehicular environments to characterize the performances of HSPA and LTE networks. For both urban and highway measurements, handover scenarios have been considered, and artificial handover traces have been created by combining the measured HSPA and LTE traces. The obtained network traces have been collected in a geo-referenced bandwidth database for further use.

Second, a vehicular DASH architecture has been proposed to upstream videos from the front facing camera of a vehicle. A simulation framework has been developed to test the performance of three AHS client rate adaptation algorithms from Liu et al. [LBG11], Miller et al. [MQGW12], and Tian et al. [TL13]. To this end, the collected network traces from the geo-referenced bandwidth database have been employed to emulate the TCP uplink throughput process during the streaming session. Besides that, a perceptual rate distortion model has been developed to take the perceptual video quality into account while determining the video levels of the full set and while evaluating the performance of the rate adaptation algorithms.

It has been shown that the selected video levels are highly dependent on the characteristics of the individual client rate adaptation algorithms. Furthermore, it has been observed that the subjective quality (measured by MOS) depends highly on the temporal and spatial activity values of the considered videos, i.e., it is content-dependent.

Finally, as the main contribution, three encoder side video level selection algorithms have been proposed to reduce the number of video levels that need to be encoded at the vehicle side. It is well known that ECUs of a vehicle have less computational capacity than CDN

servers. Therefore, encoding the same number of video levels in parallel as CDNs (10-15 in typical deployments) might exceed the computational capacity of the ECUs of a vehicle. To mitigate this, two *network-aware* and one client *request history-based* video level selection algorithms have been developed which filter the full video level set at the encoder side and create a reduced video level set. These encoder side video level filtering algorithms can be deployed on top of the standard client rate adaptation algorithms, and adjust the video level set presented to the client according to either TCP uplink throughput information or video level request history of the client, at each server side adaptation decision.

It has been shown that the proposed network-aware and client request history-based algorithms lead to a significant reduction of the number of video levels that need to be encoded while achieving a similar QoE compared to a reference implementation which considers the full set of video levels. The reduction is 83% (from 12 down to 2 levels) for both of the network-aware algorithms, and 67% (from 12 down to 2 levels) for the client request history-based algorithm. QoE is evaluated in terms of mean subjective quality of the delivered video segments, interrupted playback duration due to stalling events, and number of quality switches.

7.2 Further research

With the expected growth of adaptive HTTP streaming and its promising applications in automotive environments, video quality adaptation schemes remain a key challenge to ensure high user satisfaction. Future work can improve on some of the limitations of this thesis and extend it in several directions:

1. This thesis focuses on uplink DASH streaming in a unicast scenario where it is assumed that video is streamed in the uplink direction to a single DASH client. However, real world applications might also require multicast video delivery. For example, video from the front-facing camera of a vehicle can be streamed simultaneously to multiple entities such as other vehicles, smartphones, and off-board servers. In this case, behaviour of the adaptation algorithms might be profoundly different since multiple clients compete for the available TCP throughput. Therefore, future implementations should evaluate the performance of the adaptation algorithms in a multicast video streaming scenario.

   Another option would be to employ a two-step transmission of the video stream [LS13]. In this case, video portal can be used as an intermediate hop for both live and on-demand streaming. First, the video is upstreamed to a video portal by means of unicast transmission. Second, a streaming application server, such as a conventional CDN-based DASH system, prepares and distributes the video stream by means of multicast transmission. Thus, streaming clients do not have to share the uplink bandwidth of the source (vehicle) as in the first scenario. This in turn leads to transmission of higher video levels for each client and thus better QoE.

2. In the present thesis, it is assumed that only the video source (vehicle) is mobile, i.e., the sink (DASH client) is assumed be a stationary entity. However, in a real-world vehicle-to-vehicle uplink video streaming scenario, the DASH client is mobile and its mobility might play an important role in terms of the achievable network capacity. In this case, the bottleneck for the overall end-to-end network capacity is not necessarily determined by the uplink HetRAN (cf. Eq. (3.16)) since the average transmission capacity of the downlink HetRAN might be lower due to various degrading effects of mobility in a wireless channel.

3. Another important issue for the proposed video level selection algorithms is the reliability of the geo-referenced network trace database. In the present thesis, a representative network trace database is constructed through a small number of TCP throughput measurements. However, to guarantee the reliability of network statistics, more measurements need to be made and a larger database needs to be created. Furthermore, guaranteeing the consistency of the measurements is also difficult. For various reasons, GPS data may be inaccurate or throughput measurements may be corrupted. Hence, an automatic framework needs to be developed to filter out the inaccurate or corrupted data.

4. Vehicles connect to different HetRANs in a real-world video streaming scenario, i.e., frequent handovers between different HetRANs occur. The handover decision algorithms might be based on the received signal strength (RSS) [HVRZ05], fuzzy logic [TKW12], or other advanced techniques using cooperative information [ELO+13]. In the present thesis, all network performance measurements are made using a single RAN throughout the streaming session. Due to technical feasibility, handover scenarios are created retroactively after all the measurements are collected, assuming that handovers occur periodically (Section 4.1.2). In future implementations, real handovers performed by the HetRAN modem of the vehicle should be taken into account while creating the proposed network bandwidth database.

5. In the present thesis, on-board components of a vehicular streaming system are defined, and roles of the individual components in a DASH-based architecture are described (Section 3.4.2). However, it is not investigated how the proposed architecture should be implemented in a real automotive environment. Future research is needed to develop the implementation concepts of DASH in a vehicle. To this end, the required hardware and software building blocks need to be thoroughly defined.

Appendix A

Subjective test methodology

In the present thesis, Subjective Assessment Of Multimedia Video Quality (SAMVIQ) is used to assess the subjective quality of various automotive videos. SAMVIQ [IR08] is a subjective evaluation methodology which is specifically designed for multimedia content. It allows the individual assessors to start and stop the evaluation process or repeat the playout as they wish and determine their own speed for grading. Each scene includes an explicit reference, a hidden reference, and a number of distorted versions of the original video with various spatial and temporal impairments. The explicit reference is the uncompressed version of the video sequence and allows the assessor to determine a measure of video quality. A hidden reference, which is technically identical to the explicit reference, is inserted into the playout sequence. It is expected that the assessor identifies the hidden reference and grades it similarly to the explicit reference [IR08].

The subjective test was conducted at BMW Forschung und Technik GmbH with the participation of 17 BMW employees. For the subjective test, each of the three selected 10 s long road videos were encoded at four different frame rates (30 fps, 15 fps, 10 fps, 5 fps) and three different PSNR levels (42 dB, 38 dB, 34 dB) in H.264/AVC main profile at a constant QP using x264. Hence, for each uncompressed source video, 12 PVSs were presented to the assessors. Figure A.1 shows the employed SAMVIQ interface. Before the start of the subjective test, the following introduction was presented to the participants:

Subjective Quality of Experience Test

Within the current investigation, a novel objective video quality model to determine the perceived quality of videos from the front-facing camera of the vehicle based on the user experience is investigated. In order to validate this novel objective model, it is necessary to perform the present subjective test. The model is being developed for the use case Real Time Street View (RTSV).

By using this application the users will be allowed to identify in real time the traffic

situation from different parts of the city through the front facing camera of other vehicles. The goal of the present subjective test is to evaluate ***how good a person is satisfied with the quality of a video***, when certain video characteristics are modified. Please keep in mind that the quality of the original video does not have perfect characteristics in all cases.

Based on the results, an objective QoE model will be developed, tested and validated.

Figure A.1: SAMVIQ interface used in the subjective test.

SAMVIQ Guidelines

- The test will last approximately 1 hour. It is subdivided into two tests of approximately 25 minutes and a short break in-between.
- In each test, 14 different scenes will be evaluated within this test. For each scene an explicit reference video will be shown along with 12 processed video sequences (PVSs) of the same scene and one hidden reference.
- All these sequences from the same scene must be scored before the viewer can proceed to the next scene or previous scene.
- The interface presents a set of buttons that allow to view each of the sequences, one at time, in the video window. The duration of each sequence is 10 seconds.
- On the right hand side of the video window, there is an interactive slide-bar. Based on the task mentioned above, the quality rating can be graded from 1 to 5 with intermediate steps of 0.5 (Bad, Poor, Fair, Good, Excellent), by using the scoring slider.
- The button with label REF identifies the reference sequence. Buttons with letter labels A to M give access to either the hidden reference or one of the processed sequences.
- Moving back to previous PVSs recalls all the previous ratings.

List of Figures

List of Tables

Acronyms

ADAS	Advanced Driver Assistance Services
AHS	Adaptive HTTP Streaming
ALS	Apple Live Streaming
ANOVA	Analysis of Variance
ATM	Advanced Telecommunications Module
BDC	Body Domain Controller
BL	Base Layer
CDN	Content Delivery Network
CID	Central Information Display
CSTVQM	Contextual STVQM
DASH	Dynamic Adaptive Streaming over HTTP
DMOS	Differential Mean Opinion Score
ECU	Electronics Control Unit
EL	Enhancement Layer
ETSI	European Telecommunications Standards Institute
fps	Frames per Second
GoP	Group of Pictures
HDS	Adobe HTTP Dynamic Streaming
HSPA	High Speed Packet Access
HTTP	Hypertext Transfer Protocol
HU	Head Unit
HVS	Human Visual System
ITU	International Telecommunication Union
KPI	Key Performance Indicator
LTE	Long Term Evolution
MBR	Multiple Bitrate coding
MOS	Mean Opinion Score
MPD	Media Presentation Description
MPEG	Moving Pictures Expert Group
MSE	Mean Squared Error
MSS	Microsoft Smooth Streaming
NAT	Network Address Translation

PC	Pearson Correlation
PSNR	Peak Signal-to-Noise Ratio
PVS	Processed Video Sequence
QoE	Quality of Experience
QoS	Quality of Service
QP	Quantization Parameter
R-D	Rate-Distortion
RAN	Radio Access Network
RMSE	Root-mean-square Error
ROS	Robot Operating System
RTCP	RTP Control Protocol
RTP	Real-time Transport Protocol
RTSP	Real-time Streaming Protocol
RTT	Round Trip Time
SA	Spatial Activity
SAMVIQ	Subjective Assessment of Multimedia Video Quality
SFT	Segment Fetch Time
SRC	Source Video Sequence
SSIM	Structural Similarity Index
STVQM	Spatio-Temporal Video Quality Metric
SVC	Scalable Video Coding
TA	Temporal Activity
TCP	Transmission Control Protocol
UDP	User Datagram Protocol
URL	Universal Resource Locator
VQM	Video Quality Metric
WAN	Wide Area Network
WiMAX	Worldwide Interoperability for Microwave Access
WLAN	Wireless Local Area Network
XML	Extensible Markup Language

Bibliography

[3GP11] 3GPP. Transparent end-to-end packet-switched streaming service (PSS); progressive download and Dynamic Adaptive Streaming over HTTP (3GP-DASH, release 10) TS 26.247. Standard, 3rd Generation Partnership Project (3GPP), Valbonne, FR, November 2011.

[ABD11] Saamer Akhshabi, Ali C Begen, and Constantine Dovrolis. An experimental evaluation of rate-adaptation algorithms in adaptive streaming over HTTP. In *Proceedings of the second annual ACM conference on Multimedia systems*, pages 157–168. ACM, 2011.

[Ado10] Adobe. HTTP dynamic streaming on the Adobe Flash platform, 2010.

[ART+13] Claudio Alberti, Daniele Renzi, Christian Timmerer, Christopher Müller, Stefan Lederer, Stefano Battista, and Marco Mattavelli. Automated QoE evaluation of dynamic adaptive streaming over HTTP. In *Quality of Multimedia Experience (QoMEX), 2013 Fifth International Workshop on*, pages 58–63. Ieee, 2013.

[ATW02] John G Apostolopoulos, Wai-tian Tan, and Susie J Wee. Video streaming: Concepts, algorithms, and systems. Technical Report 260, HP Laboratories, 2002.

[BAB11] Ali C Begen, Tankut Akgul, and Mark Baugher. Watching video over the web: Part 1: Streaming protocols. *Internet Computing, IEEE*, 15(2):54–63, 2011.

[BVJS14] E. Belyaev, A. Vinel, M. Jonsson, and K. Sjoberg. Live video streaming in IEEE 802.11p vehicular networks: Demonstration of an automotive surveillance application. In *Computer Communications Workshops (INFOCOM WKSHPS), 2014 IEEE Conference on*, pages 131–132, April 2014.

[Cis14] Cisco. Cisco visual networking index: Forecast and methodology, 2013-2018, 2014.

[CMP11] Luca De Cicco, Saverio Mascolo, and Vittorio Palmisano. Feedback control for adaptive live video streaming. In *Proceedings of the Second Annual ACM*

Conference on Multimedia Systems, MMSys '11, pages 145–156, New York, NY, USA, 2011. ACM.

[CSRK11] Shyamprasad Chikkerur, Vijay Sundaram, Martin Reisslein, and Lina J Karam. Objective video quality assessment methods: A classification, review, and performance comparison. *Broadcasting, IEEE Transactions on*, 57(2):165–182, 2011.

[EKR+14] Kristian Evensen, Tomas Kupka, Haakon Riiser, Pengpeng Ni, Ragnhild Eg, Carsten Griwodz, and Pål Halvorsen. Adaptive media streaming to mobile devices: Challenges, enhancements, and recommendations. *Advances in Multimedia*, 2014, 2014.

[ELO+13] L. Ekiz, C. Lottermann, D. Ohmann, T. Tran, O. Klemp, C. Wietfeld, and C.F. Mecklenbrauker. Potential of cooperative information for vertical handover decision algorithms. In *Intelligent Transportation Systems - (ITSC), 2013 16th International IEEE Conference on*, pages 455–460, Oct 2013.

[ESS+13] A. El Essaili, D. Schroeder, D. Staehle, M. Shehada, W. Kellerer, and E. Steinbach. Quality-of-experience driven adaptive HTTP media delivery. In *Communications (ICC), 2013 IEEE International Conference on*, pages 2480–2485, June 2013.

[Ess14] Ali El Essaili. *Towards User-centric Video Transmission in Next Generation Mobile Networks*. Dissertation, Technical University of Munich, Munich, 2014.

[ETKM13] L. Ekiz, A Thiel, O. Klemp, and C.F. Mecklenbrauker. MIMO performance evaluation of automotive qualified LTE antennas. In *European Conference on Antennas and Propagation*, pages 1412–1416, April 2013.

[ETS10] TR ETSI. Human factors: Quality of experience (QoE) requirements for real-time communication services. Technical report, ETSI, December 2010.

[FGM+99] R. Fielding, J. Gettys, J. Mogul, H. Frystyk, L. Masinter, P. Leach, and T. Berners-Lee. Hypertext Transfer Protocol – HTTP/1.1. RFC 2616 (Draft Standard), June 1999. Obsoleted by RFCs 7230, 7231, 7232, 7233, 7234, 7235, updated by RFCs 2817, 5785, 6266, 6585.

[FSWV07] Rosario Feghali, Filippo Speranza, Demin Wang, and Andre Vincent. Video quality metric for bit rate control via joint adjustment of quantization and frame rate. *Broadcasting, IEEE Transactions on*, 53(1):441–446, 2007.

[GGKV98] Bernd Girod, Robert M Gray, Jelena Kovacevic, and Martin Vetterli. Image and video coding. *part of 'The past, present, and future of image and multidimensional signal processing' in Signal Proc. Mag., R. Chellappa, B. Girod, DC Munson, Jr., AM Tekalp, and M. Vetterli, Eds*, pages 40–46, 1998.

[Gir93] Bernd Girod. What's wrong with mean-squared error? In *Digital images and human vision*, pages 207–220. MIT press, 1993.

[GOMF12] P. Gomes, C. Olaverri-Monreal, and M. Ferreira. Making Vehicles Transparent Through V2V Video Streaming. *IEEE Transactions on Intelligent Transportation Systems*, 13(2):930–938, 2012.

[GTH+13] M. Grafl, C. Timmerer, H. Hellwagner, D. Negru, W. Cherif, and S. Battista. Scalable video coding guidelines and performance evaluations for adaptive media delivery of high definition content. In *Computers and Communications (ISCC), 2013 IEEE Symposium on*, pages 000855–000861, July 2013.

[H+52] David A Huffman et al. A method for the construction of minimum redundancy codes. *proc. IRE*, 40(9):1098–1101, 1952.

[HE14] Philippe Hanhart and Touradj Ebrahimi. Calculation of average coding efficiency based on subjective quality scores. *Journal of Visual Communication and Image Representation*, 25(3):555 – 564, 2014. QoE in 2D/3D Video Systems.

[HFBS09] I Herranz, S. Fikar, E. Biebl, and AL. Scholtz. Automotive multi-standard RF front-end for GSM, WCDMA and mobile WiMAX. In *Wireless Telecommunications Symposium, 2009. WTS 2009*, pages 1–5, April 2009.

[HHH+12] Te-Yuan Huang, Nikhil Handigol, Brandon Heller, Nick McKeown, and Ramesh Johari. Confused, timid, and unstable: picking a video streaming rate is hard. In *Proceedings of the 2012 ACM conference on Internet measurement conference*, pages 225–238. ACM, 2012.

[HJM13] Te-Yuan Huang, Ramesh Johari, and Nick McKeown. Downton abbey without the hiccups: Buffer-based rate adaptation for HTTP video streaming. In *Proceedings of the 2013 ACM SIGCOMM Workshop on Future Human-centric Multimedia Networking*, FhMN '13, pages 9–14, New York, NY, USA, 2013. ACM.

[HTG08] Quan Huynh-Thu and Mohammed Ghanbari. Scope of validity of PSNR in image/video quality assessment. *Electronics letters*, 44(13):800–801, 2008.

[HVRZ05] M.N. Halgamuge, Hai Le Vu, K. Rarnamohanarao, and M. Zukerman. Signal-based evaluation of handoff algorithms. *Communications Letters, IEEE*, 9(9):790–792, Sep 2005.

[IR96] ITU-R. P-800: Methods for subjective determination of transmission quality. Recommendation, ITU, Geneva, CH, 1996.

[IR07] ITU-R. Methodology for the subjective assessment of video quality in multimedia applications. Recommendation, ITU, Geneva, CH, 2007.

[IR08] ITU-R. BT.910: Subjective video quality assessment methods for multimedia applications. Recommendation, ITU, Geneva, CH, 2008.

[ISO93] Coding of moving pictures and associated audio for digital storage media at up to about 1.5 Mbit/s, MPEG-1. Standard, International Organization for Standardization, Geneva, CH, 1993.

[ISO00] Information technology – generic coding of moving pictures and associated audio information: Video. Standard, International Organization for Standardization, Geneva, CH, 2000.

[ISO03] Coding of audio visual objects – part 10: Advanced Video Coding. Standard, International Organization for Standardization, Geneva, CH, May 2003.

[ISO12] Information technology - Dynamic Adaptive Streaming over http (DASH) - part 1: Media presentation description and segment formats. Standard, International Organization for Standardization, Geneva, CH, april 2012.

[ISO13] Information technology – high efficiency coding and media delivery in heterogeneous environments – part 2: High Efficiency Video Coding. Standard, International Organization for Standardization, Geneva, CH, May 2013.

[IT13] ITU-T. Reference guide to quality of experience assessment methodologies. Recommendation G.1011, International Telecommunication Union, Geneva, CH, May 2013.

[ITU08a] ITU. ITU-T E.800: Definitions of terms related to quality of service, 2008.

[ITU08b] ITU. P. 10: Vocabulary for performance and quality of service, amendment 2: New definitions for inclusion in recommendation ITU-T P. 10/G. 100. Recommendation, ITU, Geneva, CH, 2008.

[ITU12] ITU. ITU-R BT.500-13 : Methodology for the subjective assessment of the quality of the television pictures, 2012.

[JÖ11] Dmitri Jarnikov and Tanır Özçelebi. Client intelligence for adaptive streaming solutions. *Signal Processing: Image Communication*, 26(7):378–389, 2011.

[LBG11] C. Liu, I. Bouazizi, and M. Gabbouj. Rate adaptation for adaptive HTTP streaming. In *ACM Conference on Multimedia Systems*, pages 169–174, New York, NY, USA, 2011. ACM.

[LBHG12] Chenghao Liu, Imed Bouazizi, Miska M Hannuksela, and Moncef Gabbouj. Rate adaptation for dynamic adaptive streaming over HTTP in content distribution network. *Signal Processing: Image Communication*, 27(4):288–311, 2012.

[LC12] Khalil Ur Rehman Laghari and Kay Connelly. Toward total quality of experience: A QoE model in a communication ecosystem. *Communications Magazine, IEEE*, 50(4):58–65, 2012.

[LE12] Jong-Seok Lee and T. Ebrahimi. Perceptual video compression: A survey. *Selected Topics in Signal Processing, IEEE Journal of*, 6(6):684–697, Oct 2012.

[LGSS] Christian Lottermann, Serhan Gül, Damien Schroeder, and Eckehard Steinbach. Network-aware video level encoding for uplink adaptive HTTP streaming. submitted to IEEE International Conference on Communications (ICC) 2015.

[LLS07] Yang Liu, Z.G. Li, and Yeng Chai Soh. A novel rate control scheme for low delay video communication of H.264/AVC standard. *Circuits and Systems for Video Technology, IEEE Transactions on*, 17(1):68–78, Jan 2007.

[LMS+14] Christian Lottermann, Alexander Machado, Damien Schroeder, Wolfgang Hintermaier, and Eckehard Steinbach. Camera context based estimation of spatial and temporal activity parameters for video quality metrics in automotive applications. In *IEEE International Conference on Multimedia and Expo (ICME 2014)*, 2014.

[LS13] Christian Lottermann and Eckehard Steinbach. Dienstgüteabhängige Übertragung von Medieninhalten im automobilen Kontext. Ergebnisbericht, 11 2013.

[LS14] Christian Lottermann and Eckehard Steinbach. Modeling the bit rate of H.264/AVC video encoding as a function of quantization parameter, frame rate and GoP characteristics. In *Multimedia and Expo Workshops (ICMEW), 2014 IEEE International Conference on*, pages 1–6, July 2014.

[LSS14] Christian Lottermann, Alexander , Damien Schroeder, and Eckehard Steinbach. Bit rate estimation for H.264/AVC video encoding based on temporal and spatial activities. In *IEEE International Conference on Image Processing (ICIP 2014)*, 2014.

[LZG+14] Zhi Li, Xiaoqing Zhu, Joshua Gahm, Rong Pan, Hao Hu, Ali C Begen, and David Oran. Probe and adapt: Rate adaptation for HTTP video streaming at scale. *Selected Areas in Communications, IEEE Journal on*, 32(4):719–733, 2014.

[Mac13] Alexander Machado. Modeling objective QoE and task performance metrics for automotive video streaming applications. Master's thesis, Technical University of Munich, Munich, 2013.

[MCC11] Ricky KP Mok, Edmond WW Chan, and Rocky KC Chang. Measuring the quality of experience of HTTP video streaming. In *Integrated Network Management (IM), 2011 IFIP/IEEE International Symposium on*, pages 485–492. IEEE, 2011.

[MFW13] Zhan Ma, F.C.A Fernandes, and Yao Wang. Analytical rate model for compressed video considering impacts of spatial, temporal and amplitude resolutions. In *Multimedia and Expo Workshops (ICMEW), 2013 IEEE International Conference on*, pages 1–6, July 2013.

[MLCC12] Ricky KP Mok, Xiapu Luo, Edmond WW Chan, and Rocky KC Chang. QDASH: a QoE-aware DASH system. In *Proceedings of the 3rd Multimedia Systems Conference*, pages 11–22. ACM, 2012.

[MLT12] C. Müller, S. Lederer, and C. Timmerer. An evaluation of Dynamic Adaptive Streaming over HTTP in vehicular environments. In *Workshop on Mobile Video*, MoVid '12, pages 37–42, New York, NY, USA, 2012. ACM.

[MQGW12] K. Miller, E. Quacchio, G. Gennari, and A Wolisz. Adaptation algorithm for adaptive streaming over HTTP. In *International Packet Video Workshop*, May 2012.

[MRL+12] Christopher Müller, Daniele Renzi, Stefan Lederer, Stefano Battista, and Christian Timmerer. Using scalable video coding for dynamic adaptive streaming over HTTP in mobile environments. In *Signal Processing Conference (EUSIPCO), 2012 Proceedings of the 20th European*, pages 2208–2212. IEEE, 2012.

[MTKM09] A.F. Molisch, F. Tufvesson, J. Karedal, and C.F. Mecklenbrauker. A survey on vehicle-to-vehicle propagation channels. *Wireless Communications, IEEE*, 16(6):12–22, December 2009.

[MXOW12] Zhan Ma, Meng Xu, Yen-Fu Ou, and Yao Wang. Modeling of rate and perceptual quality of compressed video as functions of frame rate and quantization stepsize and its applications. *Circuits and Systems for Video Technology, IEEE Transactions on*, 22(5):671–682, May 2012.

[NEE+11a] P. Ni, R. Eg, A Eichhorn, C. Griwodz, and P. Halvorsen. Spatial flicker effect in video scaling. In *International Workshop on Quality of Multimedia Experience*, pages 55–60, Sept 2011.

[NEE+11b] Pengpeng Ni, R. Eg, A Eichhorn, C. Griwodz, and P. Halvorsen. Spatial flicker effect in video scaling. In *Quality of Multimedia Experience (QoMEX), 2011 Third International Workshop on*, pages 55–60, Sept 2011.

[Nik95] Hrvoje Niksic. GNU wget. available from the master GNU archive site prep. ai. mit. edu, and its mirrors, 1995.

[OMLW11] Yen-Fu Ou, Zhan Ma, Tao Liu, and Yao Wang. Perceptual quality assessment of video considering both frame rate and quantization artifacts. *Circuits and Systems for Video Technology, IEEE Transactions on*, 21(3):286–298, March 2011.

[OMW09] Yen-Fu Ou, Zhan Ma, and Yao Wang. A novel quality metric for compressed video considering both frame rate and quantization artifacts. *City*, 80:100, 2009.

[OR98] Antonio Ortega and Kannan Ramchandran. Rate-distortion methods for image and video compression. *Signal Processing Magazine, IEEE*, 15(6):23–50, 1998.

[PM10] R Pantos and W May. HTTP live streaming. *Apple Inc., IETF draft*, 23, 2010.

[PS11] Y. Peng and E. Steinbach. A novel full-reference video quality metric and its application to wireless video transmission. In *IEEE International Conference on Image Processing*, pages 2517–2520, Brussels, Belgium, 2011.

[PW04] M.H. Pinson and S. Wolf. A new standardized method for objectively measuring video quality. *Broadcasting, IEEE Transactions on*, 50(3):312–322, Sept 2004.

[Rah09] Mehrnoush Rahmani. *A Resource-Efficient IP-based Network Architecture for In-Vehicle Communication*. Dissertation, Technical University of Munich, Munich, 2009.

[RBV+12] Haakon Riiser, Håkon S. Bergsaker, Paul Vigmostad, Pål Halvorsen, and Carsten Griwodz. A comparison of quality scheduling in commercial adaptive HTTP streaming solutions on a 3G network. In *Proceedings of the 4th Workshop on Mobile Video*, MoVid '12, pages 25–30, New York, NY, USA, 2012. ACM.

[REV+12] H. Riiser, T. Endestad, P. Vigmostad, C. Griwodz, and P. Halvorsen. Video streaming using a location-based bandwidth-lookup service for bitrate planning. *ACM Trans. Multimedia Comput. Commun. Appl.*, 8(3):24:1–24:19, August 2012.

[Ric04] Iain E Richardson. *H. 264 and MPEG-4 video compression: video coding for next-generation multimedia*. John Wiley & Sons, 2004.

[Rii13] Haakon Riiser. *Adaptive Bitrate Video Streaming over HTTP in Mobile Wireless Networks*. Dissertation, University of Oslo, Oslo, 2013.

[SCFJ03] H. Schulzrinne, S. Casner, R. Frederick, and V. Jacobson. RTP: A Transport Protocol for Real-Time Applications. RFC 3550 (INTERNET STANDARD), July 2003. Updated by RFCs 5506, 5761, 6051, 6222, 7022, 7160, 7164.

[SES+14] M. Seufert, S. Egger, M. Slanina, T. Zinner, T. Hossfeld, and P. Tran-Gia. A survey on quality of experience of HTTP adaptive streaming. *Communications Surveys Tutorials, IEEE*, PP(99):1–1, 2014.

[SF68] Irwin Sobel and Gary Feldman. A 3x3 isotropic gradient operator for image processing. *a talk at the Stanford Artificial Project in*, pages 271–272, 1968.

[Sha48] C.E. Shannon. A mathematical theory of communication. *Bell System Technical Journal*, 27:379–423, 623–656, 1948.

[SMW07] Heiko Schwarz, Detlev Marpe, and Thomas Wiegand. Overview of the scalable video coding extension of the H. 264/AVC standard. *Circuits and Systems for Video Technology, IEEE Transactions on*, 17(9):1103–1120, 2007.

[Sod11] I Sodagar. The MPEG-DASH standard for multimedia streaming over the internet. *MultiMedia, IEEE*, 18(4):62–67, April 2011.

[SR12] Patrick Seeling and Martin Reisslein. Video transport evaluation with H.264 video traces. *Communications Surveys Tutorials, IEEE*, 14(4):1142–1165, Fourth 2012.

[SRL98] H. Schulzrinne, A. Rao, and R. Lanphier. Real Time Streaming Protocol (RTSP). RFC 2326 (Proposed Standard), April 1998.

[Sto11] T. Stockhammer. Dynamic adaptive streaming over HTTP: Standards and design principles. In *ACM Conference on Multimedia Systems*, pages 133–144, New York, NY, USA, 2011.

[TAPS+14] L. Toni, R. Aparicio-Pardo, G. Simon, A. Blanc, and P. Frossard. Optimal set of video representations in adaptive streaming. In *ACM Conference on Multimedia Systems*, MMSys '14, pages 271–282, New York, NY, USA, 2014. ACM.

[TKW12] T. Tran, M. Kuhnert, and C. Wietfeld. Performance evaluation of feasible and holistic CSH-MU handoff solution for seamless emergency service provisioning. In *Computers and Communications (ISCC), 2012 IEEE Symposium on*, pages 000317–000324, July 2012.

[TL12] Guibin Tian and Yong Liu. Towards agile and smooth video adaptation in dynamic HTTP streaming. In *Proceedings of the 8th International Conference on Emerging Networking Experiments and Technologies*, CoNEXT '12, pages 109–120, New York, NY, USA, 2012. ACM.

[TL13] G. Tian and Y. Liu. On Adaptive HTTP Streaming to Mobile Devices. In *International Packet Video Workshop*, pages 1–8, Dec 2013.

[TQD+05] A. Tirumala, F. Qin, J. Dugan, J. Ferguson, and K. Gibbs. Iperf: The TCP/UDP bandwidth measurement tool. http://iperf.sourceforge.net/, 2005. Accessed August 09, 2014.

[VBBH14] A. Vinel, E. Belyaev, B. Bellalta, and H. Hu. Live video streaming in vehicular networks. In *Communication Technologies for Vehicles*, volume 8435

of *Lecture Notes in Computer Science*, pages 156–162. Springer International Publishing, 2014.

[WBSS04] Zhou Wang, Alan C Bovik, Hamid R Sheikh, and Eero P Simoncelli. Image quality assessment: from error visibility to structural similarity. *Image Processing, IEEE Transactions on*, 13(4):600–612, 2004.

[WKLC09] Haohong Wang, Lisimachos Kondi, Ajay Luthra, and Song Ci. *4G wireless video communications*, volume 11. John Wiley & Sons, 2009.

[WLB04] Zhou Wang, Ligang Lu, and Alan C Bovik. Video quality assessment based on structural distortion measurement. *Signal processing: Image communication*, 19(2):121–132, 2004.

[WMO09] Yao Wang, Zhan Ma, and Yen-Fu Ou. Modeling rate and perceptual quality of scalable video as functions of quantization and frame rate and its application in scalable video adaptation. In *Packet Video Workshop, 2009. PV 2009. 17th International*, pages 1–9, May 2009.

[WNC87] Ian H Witten, Radford M Neal, and John G Cleary. Arithmetic coding for data compression. *Communications of the ACM*, 30(6):520–540, 1987.

[WOZ02] Yao Wang, Jörn Ostermann, and Ya-Qin Zhang. *Video processing and communications*, volume 5. Prentice Hall Upper Saddle River, 2002.

[X26] X264. x264 project. http://www.videolan.org/developers/x264.html. Accessed August 09, 2014.

[YKHH11] Jun Yao, Salil S Kanhere, Imran Hossain, and Mahbub Hassan. Empirical evaluation of HTTP adaptive streaming under vehicular mobility. In *NETWORKING 2011*, pages 92–105. Springer, 2011.

[Zam09] Alex Zambelli. IIS smooth streaming technical overview. *Microsoft Corporation*, 3, 2009.

[ZLZG14] Chao Zhou, Chia-Wen Lin, Xinggong Zhang, and Zongming Guo. A control-theoretic approach to rate adaption for DASH over multiple content distribution servers. *Circuits and Systems for Video Technology, IEEE Transactions on*, 24(4):681–694, April 2014.