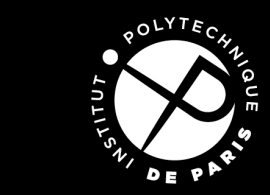


INSTITUT POLYTECHNIQUE DE PARIS

NNT : 2019IPPAT005

Thèse de doctorat

DISSERTATION

Optimization of Adaptive Video Streaming in Mobile Networks

Presented by :

THEODOROS ALEXANDROS KARAGKIOULES

In partial fulfillment of the requirements for the degree of :

Doctor of Philosophy (PhD)

Specialization in : Networking and Communication Sciences

Awarded from : Institut Polytechnique de Paris

Doctoral school : n°626 Doctoral school of Intitut Polytechnique de Paris

Prepared at : Télécom Paris and Huawei Technologies France Co. Ltd.

Defense scheduled on : 18/12/2019

Before a committee composed of :

Enrico MASALA Asc. Prof., Polytechnic of Turin	Reviewer
Gwendal SIMON Prof., IMT Atlantique	Reviewer
Rachid EL-AZOUZI Prof., University of Avignon	Examiner
Michel KIEFFER Prof., Paris-Sud University	President
Lucille SASSATELLI Asc. Prof., University of Nice, Sophia Antipolis	Examiner
Marco CAGNAZZO Prof., Télécom Paris	Director
Attilio FIANDROTTI Asc. Prof., Télécom Paris	Academic advisor
Dimitrios TSILIMANTOS Principal researcher, Huawei Technologies France Co. Ltd.	Industrial advisor

Abstract

As mobile networking technology is experiencing perpetual evolution and the computing capabilities of mobile devices are being constantly enhanced, the demand for bandwidth-intensive mobile multimedia consumption is currently experiencing an unprecedented surge. In 2017, mobile video streaming accounted for 58% of the global mobile data traffic, a percentage that is projected to reach a striking 79% by 2022 [1]. Most of this traffic is Video on Demand (VoD) [2] over HTTP Adaptive Streaming (HAS), which undoubtedly becomes fast an integral part of the mobile client's life. In order to keep pace with this explosion of video traffic, significant progress has been made in the development and design of adaptive video streaming solutions and standards. HAS solutions employ *rate adaptation algorithms*, that seamlessly adjust the rate of the media stream, to compensate for changing network conditions. Most notably, Dynamic Adaptive Streaming over HTTP (DASH) is an international standard [3] for HAS, that uses existing HTTP web server infrastructure and has now become the dominant solution for video delivery.

Nonetheless, video delivery over mobile networks still faces substantial challenges, primarily due to intrinsically unreliable channels. Mobile networks are commonly characterized by throughput variation, that is primarily attributed to physical effects associated with radio propagation, along with short-term session interruptions attributed to user mobility. Depending on the current user location, channel degradation can have a detrimental impact on user Quality of Experience (QoE).

The scope of this dissertation is to treat these challenges of video delivery in mobile networks and expedite its optimization, under two distinct perspectives. Under the perspective of Over the Top (OTT) video service providers, we propose new mobile rate adaptation algorithms; whereas under the perspective of telecommunication equipment vendors and network service operators, we explore the merits of HAS traffic collection and its analysis, on network management. Upon conducting a performance evaluation of state-of-the-art HAS schemes in mobile networks [4], we have identified 3 open issues in the current state of HAS.

The first issue concerns the type of input that drives the rate adaptation logic. Although several rate adaptation algorithms have been introduced

in order to improve user QoE, only few leverage cross-layer and sensor information that is nowadays readily available in all modern mobile devices, while most rely on either throughput estimation or application-level readings, such as the amount of pre-fetched data. In that direction, we propose a new context-aware rate adaptation solution, that incorporates cellular sensor information into the rate adaptation process.

The second open issue concerns OTT video service providers, who are continuously expanding their services to include more diverse user classes, network scenarios and streaming applications. Most existing rate adaptation algorithms depend on statistical models for the unknowns and thus face complications at generalizing appropriately well beyond a certain scope of usage. To mitigate this limitation, we propose a novel rate adaptation algorithm based on online learning, that performs well over a wide spectrum of streaming scenarios due to its design principle; its ability to learn. It does so without requiring any parameter tuning, modifications according to application type or statistical assumptions for the channel.

Last, the third open issue regards the absence of available data-sets for up-to-date mobile video streaming traffic. To better understand HAS traffic and also in order to obtain reliable data that would ultimately enable operators to optimize their networks, we conduct an extensive experimental campaign to collect cross-layer information from streaming traffic. The resulting HAS data-set is made publicly available and consists of more than 2200 real measurements (complete streaming sessions under regulated scenarios), with data recorded at the transport, network and application layer; capturing video streaming traffic over Quick UDP Internet Connections (QUIC), for the first time. In order to ensure regulated and reproducible experiments, we have developed an open source measurement application for remote and automatic control of mobile devices. Additionally, we use this data-set to design a novel traffic profiling solution, based on machine learning, that estimates parameters of HAS applications from passive measurements at the lower protocol layers.

Contents

Acronyms

AVC	Advanced Video Coding
CDN	Content Delivery Networks
DASH	Dynamic Adaptive Streaming over HTTP
HAS	HTTP Adaptive Streaming
LTE	Long Term Evolution
MDP	Markov Decision Process
MOS	Mean Opinion Score
MPD	Media Presentation Description
MPEG	Moving Picture Expert Group
NTP	Network Time Protocol
OCO	Online Convex Optimization
OTT	Over-The-Top
QoE	Quality of Experience
QoS	Quality of Service
QUIC	Quick UDP Internet Connections
TCP	Transmission Control Protocol
UDP	User Datagram Protocol
VBR	Variable Bit Rate

Chapter 1

Introduction

1.1 Motivation

Over the last decade, video delivery has evolved to constitute a major fraction of today's Internet traffic, a development that is attributed to continuous advances in networking technology, device capabilities and video compression schemes. A multitude of multimedia services such as traditional TV, Internet-Protocol TV (IPTV), video conferencing, video-on-demand and live video have rapidly converged to the streaming-via-the-Internet paradigm. Currently, video streaming accounts for more than 75% of the global Internet traffic, a percentage projected to reach a striking 82% by 2022 [5], while Over-The-Top (OTT) streaming services such as Amazon Prime, Netflix and YouTube account for more than 50% of the peak download traffic globally [2].

In parallel, the advent of powerful hand-held devices and the desire for communication on the move, have been the driving forces behind an emerging technology, known as mobile computing. Up until recently, mobile computing users were being provided with only a limited range of multimedia applications such as electronic mail, instant messaging or Voice-Over-IP (VOIP). With perpetual developments in mobile networks (4G and beyond), bandwidth intensive multimedia applications, such as high quality mobile video streaming, are currently becoming the main source of traffic that traverses through mobile networks. Cisco measured that mobile video traffic accounted for more than half (approximately 59%) of the world's mobile data traffic in 2017, while interestingly they predict that by 2022, video streaming will constitute nearly 79% of all mobile data traffic. However, there still exist significant challenges ahead, posed primarily by the intrinsic nature of mobile networks and the heterogeneity of mobile devices (e.g., CPU power, screen resolutions).

The types of multimedia applications normally supported in the wire-line networks cannot be easily supported on the mobile networks with the same quality; a limitation that is attributed mainly to their three main differences,

namely the transmission rate, error rate and mobility. For instance, the broadband wire-line network transmission links, like for instance fiber-to-the-home (FTTH) networks, are characterized by high transmission rates (in the order of Gbps) and very low error rates (10^{-9}) [6]. In contrast, mobile links, like for instance 4G-LTE, have a much smaller transmission rate (in the order of Mbps) and much higher error rate (10^{-4}) [7], primarily due to radio propagation effects such as scattering, fast fading, path loss and shadowing. Therefore, mobile networks are typically characterized by higher latency, more congestion and higher packet loss rates, that for instance in the case of the widely used Transmission Control Protocol (TCP) lead to more re-transmissions; which in turn degrade the timeliness of packet arrivals. The second major difference between the two networks is user mobility. In wire-line networks the user-network-interface remains fixed throughout the duration of a connection, whereas in a mobile environment it can keep changing, for example due to handover events, which occur when a data session is transferred to another cell. Therefore, both proper resource management and efficient network provisioning between the end-systems, are necessary to accommodate modern day multimedia services.

Multimedia traffic can be broadly classified as real-time and non-real time. Real-time traffic (e.g. video and voice) is highly delay sensitive, while non-real-time traffic (e.g. text data transfers or images) can tolerate delays. In general, video delivery is considered delay-sensitive, as data arriving too late at the client may cause severe Quality of Experience (QoE) degradation; QoE is used to measure the perception of a user's experience with a service and the factors that influence it, for the case of video streaming, are discussed in detail in Section 1.4. This real-time constraint, when combined with the inherently variable nature of mobile network environments, constitutes effective video delivery over fluctuating wireless channels an elusive task.

Currently, HTTP Adaptive Streaming (HAS) is the main technology for video streaming over the Internet, gaining significant popularity as it allows for video content to be distributed over existing web service infrastructures. In order to accommodate for bandwidth fluctuations, HAS clients can seamlessly adapt to changing network conditions by partitioning the video content and by individually controlling the download (or streaming) rate, for each part. The control logic, also called *rate adaptation*, aims at matching the video streaming rate to the channel rate. Since its inception by the DVD forum in 2002, the HAS principle has since been implemented in many proprietary solutions for video streaming, such as Microsoft's Smooth Streaming [8], Adobe's HTTP Dynamic Streaming [9] or Apples HTTP Live Streaming [10]. Therefore, in light of the wide adoption of HAS, in 2012 the Moving Picture Expert Group (MPEG) consortium created the MPEG Dynamic Adaptive Streaming over HTTP (DASH) standard [3,11], that is currently the dominant

video delivery method over HTTP[1]. DASH is completely codec-agnostic; meaning it is compatible with any codec format.

Concurrently, developments in cellular networking technology, such as 4G's Long Term Evolution (LTE), along with cheap storage for content replication and technologically advanced smartphone devices, have completely re-shaped the landscape of mobile high-quality on-demand and live video streaming services. Moreover, the anticipated cellular network evolution towards 5G, is expected to unlock even more advanced network features, that would enable mobile users to access a new suite of streaming services. Such services include, but are not limited to: Ultra High Fidelity (UHF), omni-directional (360°) mobile streaming and AR/VR applications. As HAS continues to consolidate its dominance as the main streaming technology, it is important that both the current and the next generation of streaming services are scalable and efficient over HAS, especially over the more challenging case of mobile networks.

To this end, HAS has been the subject of numerous studies and has motivated extended research effort towards its optimization, both under the multimedia and the networking principle. Principally, there exists an abundance of proposed rate adaptation algorithms, that aim at improving streaming QoE. Nonetheless, according to a performance evaluation of such algorithms in mobile networks, that we provide in Chapter 3, there still exist critical open research issues. One such issue regards the input that rate adaptation methods use in the streaming rate decision process, that is typically feedback information coming from either the network or application-layer; while HAS clients remain completely oblivious of potentially useful information coming from the other layers of the ISO/OSI stack [12]. Nonetheless, modern streaming devices, such as smartphones and tablets, are equipped with multiple sensors and instruments that enable the fusion of cross-layer readings to infer context in the mobile client's environment. In that direction, in Chapter 4, we solicit such context-awareness, by incorporating a user's inferred location into the adaptation process. Another open research issue, that this dissertation has identified, regards the use of fixed-rule schemes in place of rate adaptation algorithms, that may require parameter tuning according to the considered network or user scenario, and thus cannot generalize well beyond a certain scope of usage. Novel rate adaptation algorithms that are independent of any parameter selection concerning the network or streaming environment and that do not require computationally heavy operations, thus becoming suitable for mobile deployment, can be extremely useful to OTT service providers; who offer video streaming services directly

[1]MPEG-DASH is the only adaptive streaming over HTTP solution that is an international standard. Although the terms 'DASH' and 'HAS' are typically used interchangeably in multimedia research, in this dissertation we will distinguish between the two, by referring to 'DASH' when addressing particularly the standard and functions therein, while in relation to all other cases we will resort to the more general term 'HAS'.

to a consumer and have vested interest in enhancing user QoE in order to increase user engagement. In that direction, in Chapter 5, we present a novel rate adaptation algorithm, based on online learning, that requires no network or application modelling assumptions and is also fit for mobile deployment in terms of computational requirements.

While it is critical for service providers to maximize user QoE and optimize content distribution, the development of reliable and robust rate adaptation algorithms, although to a great extent, is only partially addressing the challenges of HAS optimization. The amendments that can be done by network operators as well, are tremendously relevant, not only to the multimedia research community, but also to the expanding streaming industry. Thus another open issue that we have identified, concerns network operators, who need detailed information of how network optimization actions may affect HAS clients and their adaptation policies. Naturally, as in any large-scale distributed system, information sharing and identification of actionable information can have a significant impact on the performance of the entire system. In the context of a network operator, by actionable information, we mean observations and measurements that can be used to take concrete actions to improve client QoE. Interestingly, HAS traffic has fundamental characteristics which can be exploited for more efficient cross-layer network management [13], specifically in mobile networks where resources are more expensive and there exist multiple user classes. Therefore, streaming traffic data-sets are a valuable resource to analyze, model and optimize network traffic and are more valuable than ever for communication research. Additionally, the commercial development of powerful new tools for monitoring and managing even encrypted HAS traffic are currently very relevant to network operators and telecommunications equipment providers. To this end, both the collection and analysis of mobile HAS traffic are treated extensively in Chapter 6, where we solicit passive traffic measurements and cross-layer traffic profiling at the cell edge, for more efficient resource management for mobile networks.

On the whole, this dissertation treats mobile multimedia delivery under two distinct perspectives. Under the perspective of OTT service providers, we evaluate existing HAS solutions and propose two new rate adaptation algorithms, in the context of *mobile* networks, according to the DASH standard. Under the perspective of telecommunication equipment vendors and mobile network operators, we explore the merits of HAS traffic collection and analysis on network management and video QoE prediction. In general, the main motivation of this dissertation is to offer novel perspectives on the optimization of mobile video delivery, precisely towards mitigating the challenges that are associated with the unpredictable and complex nature of mobile networks.

In summary, the objectives of this dissertation are:

1. to identify current challenges of existing HAS solutions
2. to propose novel rate adaptation algorithms that improve the overall performance of video streaming delivery over mobile networks
3. to provide data, tools and methods that would enable mobile network optimizations

In Chapter 1.5 we enlist our contributions towards achieving all these objectives. In parallel we provide an outline of the dissertation along with its outputs in terms of published work.

1.2 Background

Here we cover the evolution of video delivery over packet networks and we provide an extensive specification of the DASH standard and associated technologies, along with the reasons that led to its wide-spread adoption by the multimedia community.

1.2.1 Video coding technologies

A video codec is an electronic circuit or software that converts uncompressed digital video to a compressed format or vice versa. In the context of video compression, 'codec' is a concatenation of 'encoder' and 'decoder'; a device that only compresses is typically called an encoder, and one that only decompresses is a decoder. Video codecs are used in Internet video, video on demand, digital cable, digital terrestrial television, videotelephony and a variety of other applications, that record or transmit video, which may otherwise be infeasible with the high data volumes and bandwidths of uncompressed video. Currently, the most widely used video coding format is H.264 [14], also known as MPEG-4 Advanced Video Coding (AVC) [15]. This video coding standard was introduced by MPEG in collaboration with the International Telecommunication Union (ITU) and Video Coding Experts Group (VCEG).

In a motion sequence, individual frames of pictures (known as I, P and B frames) are grouped together (called a group of pictures, or GOP) and played back so that the viewer registers the video's spatial motion. An Intra (I) frame, also called a keyframe, is a single frame of digital content that the codec examines independent of the frames that precede and follow it and stores all of the data needed to display that frame. Typically, I-frames are interspersed with P-frames and B-frames (inter) in a compressed video, while the GOP structure specifies the order in which intra and inter-frames are arranged. Each coded video stream consists of successive GOPs, from which the visible frames are generated. For a given video quality, I-frames are typically 3 to 5 times larger than P-frames and up to 10 times larger

than B-frames. The role of I-frames is to provide random access points to decode the video, to allow error recovery and to allow 'fast forward' decoding, whereas P and B frames use data from lists of (already decoded) reference frames for prediction and are generally more compressible than I-frames. The difference between P and B frames, is that P frames use one single list of reference frames for prediction, while B uses two; each list can have from 1 to 16 elements. An I-frame is a complete image, while P-frames reduce the GOP size by holding only the changes in the image from the previous frame. B-frames save even more space by using differences between the current frame and both the preceding and following frames to specify the content. Therefore, I-frames contain the most amount of bits and take up more space on the storage medium (for a given video quality), whereas B and P frames are smaller [16]. This variability in encoded bits per second leads to Variable Bit Rate (VBR) video, as used by the ITU H.264 and MPEG-4 video coding standards.

An extension to AVC, known as Scalable Video Coding (SVC) [17], has been introduced that enables splitting a video stream into multiple bit-streams or 'layers', where the distinction between 'base layer' and 'refinement layers' applies. The base layer is independently decodable, while the n-th refinement layer is decodable only once the previous $n-1$ layers have been decoded, as is typically the case for all forms of scalability. SVC combines the base layer with refinement layers, in order to increment the quality of the video by increasing: the temporal resolution (or frame-rate), the spatial resolution or the Signal-to-Noise Ratio (SNR). Therefore, in regard to temporal-scalability, the video is encoded at multiple frame rates for a given resolution. The base layer has the lowest frame rate, while enhancement layers increase the frame rate, which gradually improves quality. In regard to spatial-scalability, the video is encoded at multiple spatial resolutions for a given frame rate. In case of SNR-scalability, the video is encoded at a single spatial resolution and the enhancement layers improve quality, keeping the resolution and frame-rate constant.

To achieve the same perceptual quality as AVC, by using only half the bitrate, the H.265 video codec (also known as High Efficiency Video Coding (HEVC)) was developed [18,19]. Similarly, as an extension to HEVC, Scalable High-efficiency Video Coding (SHVC) [20] was developed to support scalability. Conceptually more simple than SVC, SHVC adds extra scalability features such as bit-depth, color gamut, and hybrid scalability.

Last, royalty free encoding formats such as VP9 and AV1 have been recently introduced and are currently subject to various evaluations [21–23]. Recently, MPEG and VCEG teamed up to work on Versatile Video Coding (VVC), aiming to provide almost twice the encoding efficiency of HEVC. VVC specifically targets applications and services using immersive and high resolution video formats, such as 4K and 8K. The new standard is expected to become available in 2020 [24].

1.2.2 Transport protocols

The best-effort Internet model does not provide any mechanism for multimedia applications to reserve network resources, in order to meet the high bandwidth requirements of video streams. Furthermore, it neither prevents against uncontrolled transmissions at high rates, that can potentially cause heavy congestion in the network; leading to a congestion collapse. Therefore, it is left to the discretion of the application to dynamically adapt to network congestion.

Multimedia applications run on top of transport protocols and typically trust the transport layer to minimise induced delays and to deliver not only reliably, yet with an appropriate degree of timeliness. Originally, message-oriented transport protocols, such as Stream Control Transmission Protocol (SCTP) [25] and Datagram Congestion Control Protocol (DCCP) [26] have been thought suitable for multimedia delivery, but the existence of Network Address Translations (NAT) [27], [28], firewalls, and other middle-boxes have made their deployment challenging [29].

The Real-time Transport Protocol (RTP) [30] was proposed on top of User Datagram Protocol (UDP) for multimedia streaming. UDP uses a simple connection-less communication model with a minimum of protocol mechanisms. It provides check-sums for data integrity, and port numbers for addressing different functions at the source and destination of the datagram. It has no handshaking dialogues, and thus exposes the user's program to any unreliability of the underlying network; there is no guarantee of delivery, ordering, or duplicate protection. Therefore, as reliable data delivery and congestion control were not part of UDP's original specifications, a great amount of work has been done to augment RTP with application layer congestion control and rate control techniques, that match the rate of the video stream to the available bandwidth. Even though some techniques for robust encoding exist, like partitioning the most important data and limiting the error propagation, video streams are in general very sensitive to losses and a packet loss in an I-frame can cause serious disruptions to the video, thus both RTP congestion control and reliability remain open issues. To regulate the transmission of video packets the Web Real-Time Communication (WebRTC) framework [31], RTP multimedia congestion control [32], [33] and multipath RTP [34] mechanisms have been proposed.

TCP [35] is one of the main protocols of the Internet protocol suite. It originated in the initial network implementation in which it complemented the Internet Protocol (IP). Therefore, the entire suite is commonly referred to as TCP/IP [36]. TCP provides reliable, ordered, and error-checked delivery of a stream of octets (bytes) between applications running on hosts communicating via an IP network. Elastic applications that use TCP utilize a closed-loop feedback mechanism (built into TCP) to prevent congestion (this method of congestion control is called reactive congestion control).

Therefore, TCP is a reliable protocol which guarantees the delivery of data, but is usually associated with variable delay that stems from packet request acknowledgments and re-transmission of lost data. As video is often delay intolerant and does not always require high reliability, TCP was initially assumed unsuitable for multimedia delivery [16].

Nonetheless, in recent years TCP has become increasingly popular for multimedia delivery for a very fundamental reason. By design, TCP favors reliability to timeliness and although its congestion control tends to induce high queuing delays, it has the ability to traverse any network path that supports regular HTTP-based communication. HTTP [37] on top of TCP has become the de-facto protocol for multimedia delivery over the Internet, also widely referred to as IP-based content delivery.

Recently, Google developed QUIC [38,39] as a secure transport layer protocol, based on UDP, that aims to speed up the connections and reduce latency. QUIC enables congestion and flow control, multiple (multiplexed/pipelined) data connections (e.g., HTTP request/response) over the same UDP connection without Head-Of-Line (HOL) blocking and UDP connection migration with Forward Error Correction (FEC). QUIC can be implemented in the user space rather than the system kernel (which is the case for TCP). Therefore it can be deployed and updated more easily than TCP. To guarantee reliability, QUIC has to implement a congestion control algorithm, like Google's BBR algorithm [40].

1.2.3 Towards reliable video delivery in IP networks

Video bit-rate can only be supported on a best-effort basis, as the Internet does not provide a constant, guaranteed bandwidth for the video stream. In the case that the bandwidth is not sufficient to support the video bit-rate, then the decoder at the client receives video data at a lower rate than its consumption. Eventually that would drive the decoder to run out of video data, which in turn would result in a re-buffering event (or stall). In order to mitigate this consequence, the following solutions have been explored in an effort to match the video bit-rate to the available network throughput [16]:

- Pre-fetching data and the deployment of a play-out buffer; a queue that temporarily stores the data until they are requested by the decoder. The buffer serves as a cushion, that absorbs short-term variations in network throughput.
- Variable bit-rate compression, by appropriate parameter tuning. Such parameters include video resolution, compression ratio, or frame rate. However, this process is computationally intensive and requires complex hardware support.
- Pre-processing of video data to produce multiple encoded streams, each at a different rate, resulting in multiple versions of the same content. A rate adaptation algorithm is then used to select the most appropriate

video bit-rate given the network conditions during transmission. These solutions do not require specialized servers and use the least processing power. However, more storage and finer granularity of encoded bit-rates are required to enable the client to optimize the selection.

Considering the feasibility of deployments, the industry has settled on using play-out buffers and rate adaptation solutions. Making encoded video streams available at different bit-rates on the server, along with the mechanism that makes the appropriate video bit-rate selection (rate adaptation algorithms), have given rise to the concept of adaptive video streaming, which is the main topic of this dissertation.

1.3 Dynamic Adaptive Streaming over HTTP

Bandwidth-intensive real-time applications, such as high quality video streaming, present one of the most important challenges to the original design of the Internet. In order to support video traffic, Quality of Service (QoS) guarantees (bandwidth, delay, reliability) between the end-systems need to be provided. QoS provisioning means that video traffic should get predictable service from resources in the communication system.

In that direction, the introduction of the application layer play-out buffer to compensate for rate fluctuations, along with leveraging HTTP on top of TCP yielded a very beneficial alternative for video streaming, primarily from HTTP's ability to traverse NATs and firewalls. The initial implementation of delivering video over HTTP/TCP was called 'classic HTTP video streaming' or 'Progressive Download'; the client simply downloaded the entire video file, in a single encoded bit-rate, as fast as TCP allowed. The video player at the client-end started video playback before the download was complete. One major drawback of this technique stemmed from the fact that clients with different device capabilities and network connections would receive the same video quality, which could cause unwanted playback interruptions (stalls). This led to the development of HAS and in particular the rapid adoption of a new video streaming standard, known as Dynamic Adaptive Streaming over HTTP (DASH) [3, 41].

Briefly, DASH is an adaptive bit-rate streaming standard that enables high quality streaming of media content over the Internet, delivered from conventional HTTP web servers. According to the standard, the original video sequence is replicated in different encoding bit-rates at the server, and each encoded video stream is then divided into smaller parts, also known as segments. Thus a HAS video client can request different video bit-rates for different parts of the video sequence, according to a rate adaptation algorithm that tries to match the video bit-rate to the channel rate.

The main advantage that led to the wide spread adoption of DASH for multimedia streaming is that, with DASH the client can indirectly regulate

the server's transmission rate and due to data segmentation, there is a finer granularity of the bit-rate selection; thus better utilization of the available channel throughput. As clients make independent requests for each video segment they maintain the session state, which in turn means that clients can retrieve video segments from multiple servers. Another advantage of DASH is that it requires regular commodity web infrastructure (servers), that significantly reduces the operational costs and allows the deployment of caches to improve the performance and reduce network load [42]. In particular, proprietary commercial systems such as Microsoft's Smooth Streaming [8], Adobe's HTTP Dynamic Streaming (HDS) [9] or Apple's HTTP Live Streaming (HLS) [10] leverage existing Content Delivery Networks (CDN) and proxy caches .

In the following, we take an in-depth look at the DASH architecture, its standardization history, its operating principles, along with the main factors that affect the overall QoE of a video streaming user.

1.3.1 Architecture

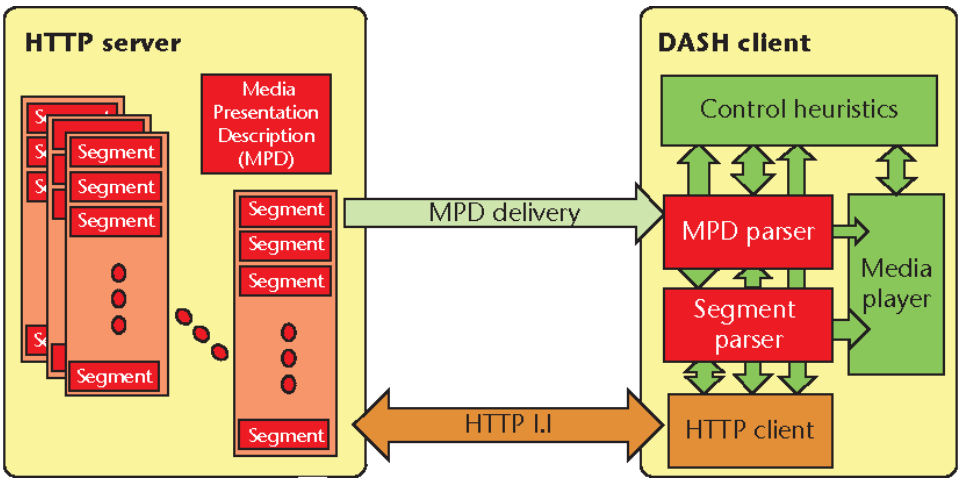


Figure 1.1: Schematic representation of the MPEG-DASH architecture (Figure from [41])

In DASH systems [11] (summarised in Figure 1.1), the video content is first encoded at multiple quality representations (e.g., multiple resolutions to meet the diverse display capabilities of different types of user devices and multiple encoding rates to facilitate adaptation to changing network characteristics) and is made available on an HTTP server. Each quality representation is organized in smaller and independently decodable files called segments; each segment typically accounting for a few seconds of video. A client desiring to access a video, initially fetches a manifest file from the server. This so called Media Presentation Description (MPD) file, contains content information such as video profiles, metadata, codecs, byte-ranges,

server IP addresses and download URLs pointing to the video segments. Video segments are 3GP-formatted and for each quality representation there is a single initialisation segment (which contains the configuration data) and many media segments. Concatenating the initialisation segment and a series of media segments results in a continuous stream of video.

Once the MPD is received and parsed, the client then deploys a control logic, also known as a *rate adaptation algorithm*, that sequentially selects the appropriate bitrate for each segment, given network conditions. Based on the bitrate indication, the client then selects the corresponding quality representation and independently requests and downloads every segment at a finite-sized queue, known as the buffer. By controlling the bitrate for each segment, rate adaptation algorithms aim at matching the video download (or streaming) rate to the channel rate. If the video consumption (or playback) rate is larger than the download rate, the buffer will deplete, eventually leading to a re-buffering event (i.e. playback interruption).

DASH does not control the video transmission rate directly, but depends on the underlying transport protocol algorithm to regulate the video transmission rate, which is determined by the congestion feedback from the client-server network path. Initially, the player starts requesting video segments sequentially as fast as possible to fill the play-out buffer; this is known as the 'filing state'. Once the buffer has been completely filled, the player enters a 'steady' state, where it periodically downloads new segments according to the decisions of the deployed control logic (rate adaptation algorithm). We treat the different states of the play-out buffer in detail in Section 6.3.3.

Interestingly, the DASH standard does not specify a particular rate adaptation algorithm. Typically, the video player observes various feedback signals such as recently achieved throughput and/or play-out buffer occupancy, that are then used as an input to the rate adaptation algorithm. The feedback is then processed, to select a suitable encoding rate (and thus the corresponding quality representation) for the download of the next segment. For example, if the throughput is high, some rate adaptation algorithms would select a higher encoding rate to provide better QoE to the user. On the other hand, if the throughput is low, the same rate adaptation algorithm would dynamically switch to a lower rate, in order to avoid play-out buffer under-run. In general, rate adaptation algorithms are responsive to fluctuating network conditions and proceed to adaptation of the encoding rate of each segment in order to provide better QoE.

The discrete set of available video representations provided by the server (contained in the MPD) and the continuous fluctuation of the network bandwidth, constitute the rate adaptation process as a discrete optimization problem where the DASH clients cannot match the network throughput perfectly; they can only achieve the (discrete) video rates described in the MPD.

1.3.2 Standardisation

MPEG-DASH has been standardised by 3GPP and became an ISO/IEC (International Electrotechnical Commission) standard for adaptive streaming in 2012 [3]. The standard defines guidelines for media presentation, segmentation, and a collection of standard XML formats for the manifest file (MPD). However, specific client implementation and rate adaptation techniques are not specified in the standard. Hence, commercial streaming services that use DASH implement their own proprietary techniques both for media representation and for client rate adaptation. Thus, an extensive research interest has spurred on rate adaptation algorithms and DASH solutions in general, that are reviewed in Chapter 2. DASH has subsequently been adopted by other standardised multimedia streaming systems such as IPTV [43] and Digital Video Broadcasting (DVB) [44], as a standard for transporting video over the Internet.

The DASH Industry Forum (DASH-IF) (a group containing leading streaming companies) drives the adoption and research in modern adaptive streaming technologies. DASH-IF provides specific implementation guidelines and regular documentation of interoperability [45]. The community has also developed 'dash.js', an open-source reference player for research and testing purposes.

1.4 Quality of Experience

Quality of Experience (QoE) is a measure of the delight or annoyance of a user's experience with a service. QoE is an emerging multidisciplinary field based on social psychology, cognitive science and engineering science. QoE is focused on understanding overall human quality requirements. The development of HAS-based technologies marked a noticeable shift from traditional QoE measurement on video (e.g., Peak Signal-to-Noise Ratio) and user experience (e.g., subjective mean opinion scores) to more complex quality metrics (e.g., re-buffering time, average video rate, rate switching frequency) and engagement-centric metrics (e.g., views per video, viewing duration). Optimizing HAS QoE [46] is a challenging goal because these metrics are often interdependent, having counter-intuitive and complex relationships. In the following, we take a closer look at the video streaming performance factors that influence QoE.

1.4.1 Adaptive video streaming performance factors

In general, video streaming QoE is measured by performance factors [47], that can be categorized into technical and perceptual factors. The perceptual influence factors are directly perceived by the end user of the application and are dependent but decoupled from the technical development. Therefore,

it is necessary to analyze also the technical influence factors, which drive the perception of the end users.

Seufert et al. [48] provide a comprehensive survey of the available QoE metrics investigated in both industry and academic research. Here we only present some of the key works on QoE measurements, while an analysis of QoE for mobile video streaming is provided by Poojary et al. [49].

Initial delay

Initial delay is built-in a multimedia streaming service as data must be first transferred through the network before being decoded and playback can begin. The practical value of the minimal achievable initial delay thus depends on the available transmission data rate and the encoder settings. Often video playback is delayed more than usually necessary in order to complete the initial filing phase of the play-out buffer. Indeed, there is a trade-off between the amount of initially buffered playtime and the risk of buffer depletion, i.e., stalling. In general, the impact of initial delay on video streaming QoE is not severe and thus, in this dissertation, it is not treated as a primary optimization objective of rate adaptation. Hoßfeld et al. [50] show that initial delay up to 16s reduce the perceived quality only marginally. As video streaming users are used to some delay before the start of the playback, they usually tolerate it if they actually intend to watch the video; as long as the duration of the video is significantly longer by order of magnitudes than the initial delay.

Stalling

Stalling events appear when video playback is stopped, because of play-out buffer under-run. Stalls are manifested when insufficient data are available in the buffer and video playback is paused, until the buffer increases above a threshold; usually in the order of a couple of segments. Here again, a trade off exists between the duration of the re-buffering event and the risk of a shortly recurring stall. Hoßfeld et al. [51] show that there is an exponential relationship between stalling parameters and Mean Opinion Score (MOS). Moreover, they find that users tolerate at most one stall per clip, as long as its duration remains in the order of a few seconds.

Stalling is the most critical factor of experience degradation and all video streaming services should avoid stalling whenever possible. Classical HTTP video streaming is strongly limited and cannot react to fluctuating network conditions other than by trading off play-out buffer size and stalling duration. In contrast, HAS is able to align the delivered video stream to the current network conditions, thereby mitigating the effects of stalling more effectively. A streaming session is typically characterized as: i) *consistent*, when stalling

events are infrequent and ii) *continuous*, when their duration is short.

Adaptation

The main difference of HTTP adaptive streaming and classical HTTP video streaming is the possibility to switch the video rate during the playback in order to adapt to the current network conditions. To make rate adaptation possible, clients need to measure their current network conditions at the edge of the network and control which data rate is suitable for the current conditions. Since on the server side, the video is split into small segments and each of them is available in different encoding rates, the rate adaptation algorithm should request the next part of the video in the appropriate rate level which is best suited, under current network conditions. The frequency of rate switching and the amplitude (absolute bit-rate difference between switches) at which it occurs, can have a significant effect on the user perceived experience. Typically, adaptation is a measure of: i) streaming *stability*, i.e a streaming session is stable when the effects of adaptation are restrained and ii) streaming *smoothness*, i.e a smooth session is characterized with moderate (in amplitude) rate changes between consecutive segments.

Average video rate

According to VBR encoding, the encoder will try to reach a target, in terms of average bit-rate, while allowing the bit-rate to vary between different parts of the video (more complex portions of the material require more bits for the encoding, while the opposite is true for less complex areas). In video streaming, the average bit-rate over all downloaded segments, is a reliable and quantitatively direct technical indicator of the quality of the stream; under the typical assumption, that compression methods use higher bit-rates to encode material at a higher quality, when comparing the same content in the same resolution.

Video Quality Assessment

The legitimate judges of visual quality are humans as end-users, and their opinions can only be obtained by subjective experiments, that involve a panel of participants which are usually non-experts, also referred to as test subjects, to assess the perceptual quality of given test material such as a sequence of videos. Subjective experiments are typically conducted in a controlled laboratory environment.

The outcomes of a subjective experiment are the individual scores given by the test subjects. These scores are used to compute MOS or Differential Mean Opinion Score (DMOS) depending on how the experiment is designed.

The main difference between MOS and DMOS, is that MOS is the outcome when the subjects rate a stimulus in isolation, while for DMOS the change in quality between two versions of the same stimulus is rated.

Due to the time-consuming nature of executing subjective experiments, large efforts have been made to develop objective quality metrics [52]. The purpose of such objective quality methods is to automatically predict MOS with high accuracy. In evaluating and comparing video quality one should effectively distinguish and compare the performance obtained using different measures and data-sets.

However the training and performance analysis of objective Video Quality Assessment (VQA) algorithms is complex due to the huge variety of possible content classes and transmission distortions. To this end, Aldahdooh et al. [53] focus on how to improve current performance evaluation measures and introduce a new VQA measure. Similarly, a full-reference VQA method, based on ensemble-learning, is proposed by Lin et al. [54]. The latest VQA metric is Netflix's Video Multi-method Assessment Fusion (VMAF) [55], that is also a full reference metric that on image metrics, together with temporal difference between consecutive frames, based on subjective tests performed by Netflix.

Note: *While video quality assessment is a well established research field in the multimedia domain, its impact on QoE is associated more with breakthroughs in video coding technology, rather than with adaptive video streaming techniques. HAS methods try to facilitate the delivery of video content, that already exists on the hosting server in predetermined quality. Undoubtedly, the encoding decisions, that would ultimately dictate the quality of the content that becomes available for streaming, is an integral part of HAS optimization. Nonetheless, rate adaptation algorithms, although at the heart of every HAS method, are not in direct control of video quality. Instead, they are designed to map existing network conditions to the most suitable encoding rate available at the hosting server. To this end, as this dissertation approaches HAS optimization primarily under the scope of rate adaptation, we will not assess the perceptual video quality directly. Instead we will resort solely to the average rate over all segments in a session, as it is a more meaningful metric to measure adaptation performance.*

Additionally, throughout this dissertation we have purposely avoided combining influence factors into composite metrics, also known as unified QoE models, for the evaluation of HAS performance. HAS provides an inter-operable solution to overcome volatile network conditions, but the degree of impact for each individual QoE factor on the total human perceived experience, is not yet well-understood. Instead, when evaluating streaming performance, we report metrics that correspond to a quantitative representation of the technical influence QoE factors, primarily associated with average streamed rate, frequency and amplitude of adaptation (i.e rate switching), and frequency and duration of re-buffering events.

1.4.2 General rate adaptation goals

The general goal of a rate adaptation algorithm is to maximize the streamed video rate, subject to:

1. avoiding playback interruptions caused by buffer under-runs and
2. minimizing video rate oscillations.

The core challenge for rate adaptation design is balancing the rate maximization goal with the two constraints and ultimately providing high QoE for end-users. In essence, a rate adaptation algorithm is an optimization solution with the objective of maximizing the video rate, while at the same time ensuring uninterrupted and stable (i.e. minimal adaptation) streaming.

A plethora of proposed algorithms exists in both scientific literature and actual industry practices, given that no particular rate adaptation algorithm is specified in the DASH standard [11]. In Chapter 2 we provide a literature review of the state-of-the-art in HAS.

1.5 Contributions

Below we provide the contributions of this dissertation, while concurrently we outline its structure.

In Chapter 1 we specify the motivation for studying HAS, particularly in mobile networks. Additionally, we provide all required background and underlying technologies for multimedia delivery via HAS. Last, we proceed to an introduction to the basics principles of adaptive video streaming and the DASH standard in particular, along with a brief discussion concerning the relevant factors that influence video streaming QoE.

In Chapter 2 we provide a comprehensive survey of the state of the art in HAS design. In particular we attempt a classification of rate adaptation algorithms according to two principles: (i) according to the architecture module that implements the rate adaptation logic and (ii) according to the dynamic that the rate adaptation algorithm may use as an input to the decision process.

In Chapter 3, we evaluate the performance of five state-of-the-art rate adaptation algorithms and make a per class comparison, based on real network traces for multiple throughput profiles. We investigate buffer-based and hybrid rate adaptation for the first time and provide insight on the merits of each class of algorithms. Furthermore, due to the increasing popularity of live streaming services, which have small buffer sizes due to the strict real-time delay requirements, we also examine the impact of the buffer size on the performance of the studied rate adaptation algorithms. According to the results of our evaluation, we provide rate adaptation implementation guidelines to designers and operators, for the right algorithmic approach

according to expected network conditions and service requirements. The main contribution of the work presented in Chapter 3, is the identification of a fundamental limitation in the design of rate adaptation methods. Relying on pre-selected designer-specific parameters or heuristic design, constitutes still a major challenge for multimedia research, when designing robust rate adaptation algorithms aiming at high QoE under changing conditions and requirements. The insights gathered from this performance evaluation, where the drivers of the work presented in Chapter 4 and Chapter 5. Chapter 3 is associated with the following publication:

- T. Karagkioules, C. Concolato, D. Tsilimantos, and S. Valentin. *A Comparative Case Study of HTTP Adaptive Streaming Algorithms in Mobile Networks*. In Proceedings of the 2017 ACM Workshop on Network and Operating Systems Support for Digital Audio and Video (NOSSDAV), pp. 1-6. **Best paper award**.

In Chapter 4, we propose context-aware rate adaptation, leveraging modern-day mobile device capabilities in accessing cross-layer information. In particular, we extend a classic rate adaptation algorithm by a Bayesian coverage estimator, incorporating inferred location information into the rate adaptation process. We present a Bayesian detector that combines signal measurements from the cellular and GNSS receiver, that result in a binary estimate (i.e., indoors or outdoors) of the user's coverage situation in real time. Then, we apply the indoors-outdoors detector to improve the performance and stability of a classic rate adaptation method. The results of our experimental evaluation clearly demonstrate that accounting for the coverage *context* of a user is a promising approach for designing robust adaptive streaming policies for mobile users. Chapter 4 is associated with the following publications:

- S. Mekki, T. Karagkioules and S. Valentin. *HTTP Adaptive Streaming with Indoors-Outdoors Detection in Mobile Networks*. In Proceedings of the 2017 IEEE Conference on Computer Communications Workshops (INFOCOM WKSHPS), pp. 671-676.

- S. Mekki, T. Karagkioules and S. Valentin. *Context-aware adaptive video streaming for mobile users*. In Proceedings of the 2017 IEEE Conference on Computer Communications Workshops (INFOCOM WKSHPS), Atlanta, GA, 2017, pp. 988-989.

In Chapter 5, we recast rate adaptation under an optimization framework and present *Learn2Adapt*, a novel rate adaptation algorithm based on online learning. In our design, we model the adaptive streaming client by a learning agent, whose objective is to maximize the average video rate of a streaming session, subject to scheduling constraints of the buffer queue and we model the channel rate evolution by an adversary, that decides the cost of each decision only after it has been taken. These assumptions along with a set of

relaxations concerning the decision space and the buffer evolution, allow us to cast the adaptive streaming optimization problem as a constrained Online Convex Optimization (OCO) problem [56]. In our trace-based simulations, our proposed method proves to be *robust*, providing consistently higher average rate when evaluated against reference state-of-the-art rate adaptation algorithms in a wide spectrum of possible network and streaming conditions. The *robustness* property of *Learn2Adapt* allows it to be classified in the small set of rate adaptation algorithms that mitigate the main limitation of existing mobile HAS approaches; the dependence on statistical models for the unknowns. This is of significant relevance in the field of modern HAS, where OTT video service providers are continuously expanding their services to include more diverse user classes, network scenarios and streaming applications. Chapter 5 is associated with the following publication:

- T. Karagkioules, G. S. Paschos, N. Liakopoulos, A. Fiandrotti, D. Tsilimantos and M. Cagnazzo, *Optimizing Adaptive Video Streaming in Mobile Networks via Online Learning.* submitted to IEEE Transactions on Multimedia. Currently **under review**, *pre-print* is accessible at [57].

In Chapter 6 we describe our public data-set for YouTube's mobile streaming client and the methods to reproduce it. In particular we provide an extensive experimental design that enables researchers to study and to reproduce how a popular adaptive streaming client reacts to variable link conditions, or compare the performance of their own rate adaptation algorithms against YouTube's adaptive streaming policy. Additionally we specify an experimental setup that allows to design controlled experiments with automatic variation of network and streaming parameters during a video session. Besides measuring network statistics, we also provide streaming variables from YouTube's native Android application. This was possible by our recently introduced *Wrapper App*, a software tool that allows remote control and monitoring of YouTube's native Android application. All of these data are generated by YouTube's native Android application and most of the recorded logs refer to QUIC traffic, enabling the research community to study this new protocol for the first time. Next, we utilize our data-set to design a new traffic profiling system that classifies the flow type and play-back buffer state of HAS traffic at the IP layer. We believe that publishing our data and tools[2] will enable the research community to better understand how to model, manage and control adaptive streaming traffic. Chapter 6 is associated with the following publications:

- T. Karagkioules, D. Tsilimantos, S. Valentin, F. Wamser, B. Zeidler, M. Seufert, F. Loh, P. Tran-Gia. *A Public Dataset for YouTube's Mobile Streaming Client.* In Proceedings of the 2018 IEEE Network Traffic Measurement and Analysis Conference (TMA), pp. 1-6.

[2]Accessible at: `http://qoecube.informatik.uni-wuerzburg.de/`

- D. Tsilimantos, T. Karagkioules, and S. Valentin. 2018. *Classifying Flows and Buffer State for YouTube's HTTP Adaptive Streaming Service in Mobile Networks.* In Proceedings of the 2018 ACM Multimedia Systems Conference (MMSys), pp. 138-149.

- D. Tsilimantos, T. Karagkioules, Amaya Nogales-Gmez and S. Valentin. *Traffic profiling for mobile video streaming.* In Proceedings of the 2017 IEEE International Conference on Communications (ICC), Paris, 2017, pp. 1-7.

- M. Seufert, B. Zeidler, F. Wamser, T. Karagkioules, D. Tsilimantos, F. Loh, P. Tran-Gia, S. Valentin. *A Wrapper for Automatic Measurements with YouTube's Native Android App.* In Proceedings of the 2018 IEEE Network Traffic Measurement and Analysis Conference (TMA), pp. 1-8.

- F. Loh, T. Karagkioules, M. Seufert, B. Zeidler, D. Tsilimantos, P. Tran-Gia, S. Valentin, F. Wamser. *Demo: A wrapper for automated measurements with YouTube's native app.* In Proceedings of the 2017 2018 IEEE/IFIP Network Operations and Management Symposium (NOMS), Taipei, 2018, pp. 1-2.

In Chapter 7 we provide concluding remarks and a summary of the work treated in in each chapter. Additionally, we identify and outline emerging research perspectives that could potentially be at the center of future research.

Chapter 2

State of the art

Video rate adaptation schemes can be broadly classified according to the module that implements the adaptation logic. According to the DASH standard [11], multimedia delivery requires a server-client architecture, thus the rate adaptation module can be hosted at either of the two components. Most of the proposed rate adaptation schemes, reside at the client-side, where rate adaptation decisions are made according to either network or application-level information, a combination of both, or even cross-layer metrics. In the following, we begin our literature review[1] with such client-side approaches, as they require the least amount of architectural assumptions and are more relevant to the proposed methods presented in the following chapters herein. We then proceed to outline network-assisted rate adaptation methods, that allow HAS clients to take network information into consideration, typically obtained by an in-network element, for optimization of the rate adaptation process. Last, we review server-side rate adaptation methods, that require no cooperation from the client, but instead resort to traffic shaping methods at the server-side alone. Similarly, complete surveys on HAS techniques are independently provided by Kua et al. [58], Sani et al. [59] and Bentaleb et al. [60].

2.1 Client-side rate adaptation

Client-side rate adaptation schemes try to adapt to throughput variations by switching to an appropriate video encoding rate according to one or more metrics that are immediately available to the client, such as the available throughput, playback buffer size, etc. Figure 2.1 shows a simple model of client-side rate adaptation. The client uses one or more metrics that depict directly (throughput) or indirectly (amount of pre-fetched data in the

[1] While this review covers a great part of the state of the art, it is not exhaustive. To this end, we complement Chapter 2 with individual 'Prior work' subsections in Chapter 3 to Chapter 6 that contain, additional material.

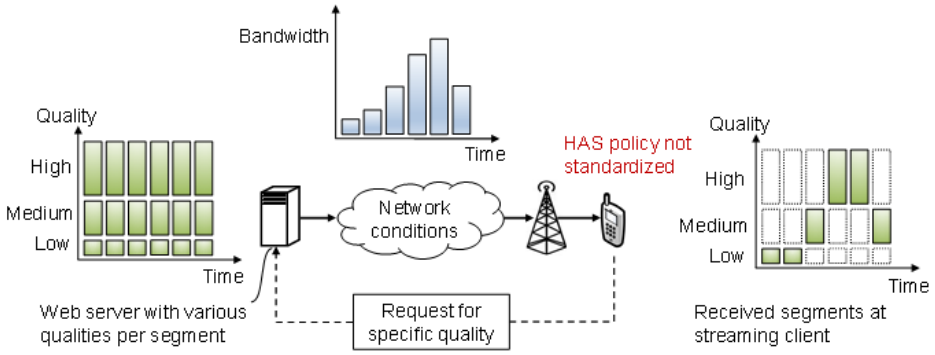


Figure 2.1: Schematic representation of the client-side rate adaptation

buffer queue) network conditions, as input in the rate selection algorithm, in order to choose the appropriate bit-rate level for the next segment download. These algorithms try to avoid streaming problems like, rate oscillations and buffer starvation, while improving viewer QoE. In particular, rate adaptation algorithms strive to achieve (i) minimal re-buffering events that occur when the playback buffer depletes, (ii) a high overall playback rate with respect to network resources and (iv) minimal video rate oscillations, which occur due to frequent rate switching.

Below, we further organize the client-based rate adaptation methods into three classes (i.e. throughput-based, buffer-based or hybrid), according to the dynamics used into the rate adaptation process. Furthermore, in Chapter 3, we treat a per class performance evaluation of client-side rate adaptation [4].

2.1.1 Throughput-based rate adaptation

According to this client-side adaptation scheme, the rate adaptation logic is based on the measured available network throughput, which is usually calculated as a function of the size of the fetched segments and the actual download time.

Li et al. [61] propose *PANDA*, a notable throughput-based rate adaptation method based on a four-step model, where initially the available network throughput is estimated using a proactive probing mechanism, that is designed to minimize video rate oscillations. Then, the throughput estimates are smoothed using noise-filters to avoid estimation errors due to throughput variation and a decision of the encoding rate for the next segment is made. In the last step, the scheduling of the download is considered with the objective to drive the buffer level towards the maximum buffer level. Typically, in throughput-based adaptation, the inter-request time between consecutive segments is matched to the necessary time needed to complete the download, based on the smoothed estimated available throughput.

Liu et al. [62], propose a rate adaptation algorithm that detects through-

put fluctuations and congestion using a filter on the download duration, which is defined by the start of the HTTP GET request until the receipt of the last byte of the segment. In a similar approach Liu et al. [63], include both sequential and parallel segment fetching methods in CDNs, by comparing the expected download time with the measured download time, to determine if the selected segment bit-rate matches the network capacity. In general, download time indirectly depicts the available throughput and can vary as a function of the segment size, that in the case of VBR encoding can vary according to the content characteristics. Thus, download time is considered as a higher level parameter than throughput and some works [4] have even treated it as a separate class of client-side adaptation, namely *time-based* adaptation.

When competing for a bottleneck link, fairness amongst multiple clients is one of the most significant QoS factors related to video streaming. In that direction, a rate adaptation strategy from Jiang et al., called *FESTIVE* [64], focuses primarily on fairness amongst all clients and studies the limitations of video players when a large number of clients share the same network.

In the nature of live streaming, real-time constraints apply. Miller et al. [65] propose a low-latency prediction based rate adaptation scheme over wireless access links, which leverages TCP throughput predictions to achieve low latency and improve viewer QoE.

In the context of live streaming as well, Xiao et al. propose DASH to Mobile (DASH2M) [66] as a strategy with the goal of enhancing the QoE as well as reducing the battery consumption of the mobile client using HTTP/2 server push and stream termination properties.

HTTP/2's stream resetting feature has been also leveraged by Yahia et al. [67], who propose methods that insure very low delivery latency by applying video frame discarding policies inside a video segment, when a selected DASH representation does not match with the available network resources.

2.1.2 Buffer-based rate adaptation

In this type of client-side rate adaptation scheme, the client uses application-level signals, such as the instantaneous buffer level, to perform the rate adaptation.

BBA, one of the most notable buffer-based rate selection algorithms, proposed by Huang et al. [68], aims to maximize the average video rate and avoid unnecessary re-buffering events using a direct mapping of the instantaneous buffer level to the set of available rate levels, or as function of the segment size.

Another significant contribution in buffer-based rate adaptation, is *BOLA* [69], an approach that is based on control theory and in particular Lyapunov optimization [70]. *BOLA*, is an online control algorithm that aims to maxi-

mize a utility that is associated with the average bit-rate and re-buffering time, while adapting to network changes to account for better QoE. *BOLA* is implemented in the DASH-IF reference player [71].

Yadav et al. [72] model a DASH client as an M/D/1/K queue referred to as a *QUETRA*, which allows to calculate the expected buffer occupancy, given a bit-rate choice, network throughput, and buffer capacity.

A similar approach based on queuing theory [73] as well, is provided by Burger et al. [74], who model the video buffer as GI/GI/1 queue with pq-policy using discrete-time analysis. By studying the stochastic properties of the buffer-level distribution, the authors evaluate accurately the impact of network and video bit-rate dynamics on the video playback quality.

2.1.3 Hybrid rate adaptation

This class of rate adaptation schemes is not limited to only network or application-level information. Hybrid rate adaptation algorithms may resort to a combination of inputs, that may come from any layer of the OSI model [12], or may even include video content characteristics.

In that direction, Miller et al. [75], combine the current buffer occupancy level, estimated available throughput, and average bit-rate of the different bit-rate levels from the MPD, as input metrics in the video rate selection. The algorithm is dynamic in the sense that it changes its behavior based on the current buffer level while aiming to accurately estimate the available throughput and avoid throughput overestimation and numerous rate oscillations, and playback interruptions.

Juluri et al. [76] propose *SARA*, that is a segment-aware rate adaptation algorithm that is based on the segment size variation, the available throughput estimate and the instantaneous buffer occupancy. In particular, the authors propose to enhance the typical MPD file to include the size of every segment and for each new segment download, the client decides the new segment rate based on the estimated throughput (which is assessed using the segment size) and the current status of the buffer.

In mobile networks, users may commonly lose coverage when entering a building due to the high signal attenuation at windows and walls. Under such conditions, services with minimum bit-rate requirements, such as video streaming, often show poor QoE. Mekki et al. [77] present a Bayesian detector that combines measurements from two smartphone sensors to decide if a user is inside a building or not. The proposed algorithm is a cross-layer context-aware rate adaptation algorithm. More details are provided in Chapter 4.

Toni et al. [78] propose guidelines for optimal selection of the set of available representations, taking into account network dynamics, type of video content, and user population characteristics aiming to provide fairness among users and to reduce network resource usage.

ABMA+ [79] is a lightweight rate adaptation algorithm that selects the highest quality representation based on the estimated probability of video re-buffering. It relies on the principle of probing, but this time to estimate the download time of each segment. *ABMA+* is a hybrid adaptation method, in the sense that it is a time-based adaptation method that also makes use of buffer maps, which define the play-out buffer capacity that is required under certain conditions to satisfy a re-buffering threshold and to avoid heavy online calculations.

Some rate adaptation methods may resort to optimization and control techniques [80], where the video streaming process is typically formulated through a utility function. Streaming agents invoke learning techniques in order to make rate adaptation decisions under fluctuating network conditions. In that direction, Bokani et al. [81] model the rate adaptation logic as a Markov Decision Process (MDP), and incorporate mobility by including vehicular environments. The algorithm takes advantage of the historical throughput samples to build an accurate throughput estimation model. Similarly, Bao et al. [82] present a solution that adapts to the current and predicted channel state as well. Their formulation is based on MDP with real content and channel traces from vehicular users.

A real-time best-action search algorithm over multiple access networks was proposed by Xing et al. in [83], where Bluetooth and WiFi links were used simultaneously to download video segments. An MDP is formulated so that the rate adaptation agent takes the buffer level, SVC layer index, Bluetooth traffic, available throughput, and the index of each segment fetched as inputs. The reward function is designed to consider the average playback quality, interruption rate, and playback smoothness.

Bentaleb et al. [84] discuss the shortcomings of the existing client-based schemes and leverage a game theory framework to sidestep these drawbacks. The authors use a cooperative game in coalition form and then formulate the bit-rate selection problem as a bargaining process and consensus mechanism. Thus, the DASH clients can create an agreement among themselves and achieve their QoE objectives.

Changuel et al. [85] propose a cross-layer control [86] mechanism to stream efficiently scalable videos to mobile receivers, with the goal to maximize the average rate of the received video while accounting for the variations of the characteristics of the transmitted content and of the channel. The control problem is cast in the framework of MDP, while the optimal actions to apply to the system are learned using reinforcement learning [87].

Chiariotti et al. [88] propose an online bit-rate adaption algorithm for DASH clients that casts an MDP optimization for the selection of the optimal representations. System dynamics required in the MDP model are a priori unknown and are therefore learned through a reinforcement learning technique. The developed learning process exploits a parallel learning technique that improves the learning rate and limits sub-optimal choices,

leading to a fast and yet accurate learning process that quickly converges to high and stable rewards.

In Chapter 5, we recast the rate adaptation optimization problem under the view of online convex optimization [56] and present a novel algorithm [57] for video rate adaptation, named *Learn2Adapt*. The proposed algorithm is shown to provide a *robust* rate adaptation strategy which, unlike most of the state-of-the-art techniques, does not require parameter tuning, channel model assumptions, or application-specific adjustments and does not require computationally heavy implementations. These properties make it very suitable for mobile users, who typically experience fast variations in channel characteristics or limited processing capabilities issues.

2.2 Network-assisted rate adaptation

The network-assisted approach allows the collection of measurements about the network conditions while assisting clients in their selection of the appropriate streaming rate. A special component (e.g., agent/proxy deployed in the network) is typically required to monitor the network status and conditions, that allows DASH clients to efficiently use network resources.

Mok et al. propose *QoE-aware DASH (QDASH)* [89], that integrates available throughput measurement into the video data probes with a measurement proxy architecture. Mok et al. found that the available throughput measurement method facilitates the selection of video rate levels. QDASH consists of two components: (i) a module to measure the throughput and (ii) a module that assists the client in choosing a suitable bit-rate.

Similarly, Bouten et al. [90] propose a QoE-driven in-network optimization system for adaptive video streaming, that consists of a set of agents deployed along the path between the clients and streaming server, as proxies. These agents periodically measure and monitor the available throughput along the path using packet sampling techniques and solve an optimization problem to determine the optimal bit-rate for the next segments to be downloaded.

Most network-assisted solutions rely on the assumption that network elements have full knowledge of the network status, which is not always realistic. D'Aronco et al. [91] propose a distributed price-based network-assisted DASH system where the network elements infer the network link congestion using measurements collected from the client-end. The congestion level signal is then used by the clients to optimize their video data requests.

The recent introduction of CDN has improved significantly the quality of services delivering content to end users. Gourdin et al. in [92] analyze the impact of a revenue-maximizing CDN on the end-users, the network operators and the content providers. In an effort to reduce the delivery (traffic) costs of a CDN, He et al. [93] propose a mechanism that reduces the number of segments that need to be encoded and delivered, by tracking

'free-riding' live streaming users.

Information Centric Networks (ICN) [94] are a data-centric communication paradigm [95] that leverages a content-aware connectionless transport, including network-level caching, multi-path forwarding capabilities and seamless mobility support; features that are all very appealing for HAS systems. Samain et al. [96] compare ICN and TCP/IP with an experimental approach, employing several state-of-the-art HAS algorithms on an ICN. In the same context, Samain et al. [97] propose and evaluate simple signals coming from in-network telemetry that are effective to enhance the quality of HAS streaming.

Thomas et al. [98] propose *SAND*, a control plane that offers asynchronous client-to-network, network-to-client, and network-to-network communications. SAND allows to collect metrics and status information from different entities in the system including the clients and to send feedback to the clients and network elements including the servers, caches and other network entities along the media path. Jmal et al. [99] study how the SAND architecture can address the problems met by the delivery systems that implement cache proxies and propose a novel delivery architecture, which combines SAND and Content Centric Networks (CCN) [100].

Han et al. [101] propose Multi-path DASH (MP-DASH), a multi-path framework with awareness of the network interface preferences of the clients. It aims to improve multi-path TCP (MPTCP), the merits of which were investigated also by James et al. [102] and Poliakov et al. [103], to support DASH considering the user network interface preferences, thus enhancing the efficiency of video delivery without sacrificing viewer QoE.

An investigation of several network-assisted streaming approaches, along with a comparison with client-side rate adaptation algorithms can be found by Cofano et al. [104]. Network-assisted approaches, may generate significant overhead in the network, especially with increasing number of clients. This overhead may eventually lead to network congestion, resulting in a low QoE.

2.3 Server-side rate adaptation

Server-based schemes rely on bit-rate shaping methods at the server side and do not require any cooperation from the client. The switching between the bit-rates, therefore, is implicitly controlled by the server bit-rate shaping mechanism.

Akhshabi et al. [105] deploy a traffic shaping method at a home gateway to improve fairness, stability and convergence delay and to eliminate the OFF periods during the steady states, which is the root cause of the instability problem.

Houdaille et al. [106] propose a similar mechanism that allows bandwidth arbitration to be implemented in the home gateway, first determining desir-

able target bit-rates to be reached by each stream and then constraining the clients to stay within their limits.

In regard to live streaming, Detti et al. [107] propose an architecture that consists of clients communicating with a server through a shared proxy and a server having a tracker functionality that manages the clients' states and helps in the communication.

In the same context of live streaming, De Cicco et al. [108] propose a feedback mechanism, which aims to control the size of the buffer, on the server side, in order to adjust and select the most appropriate bit-rate level for each HAS client.

Multi-user wireless networks, where streaming clients constantly adapt the rate of their video streams, are commonly characterized by network overload and rapid bandwidth variation; which in turn results to bit-rate instability. Thus, to accommodate the dynamic nature of streaming traffic, most resource allocation schemes require close coordination between content providers, clients and network elements. This condition is often infeasible due to scalability and network operator policies. Recently, Sunny et al. [109] presented *D-VIEWS*; a scheduling paradigm that assures video bit-rate stability of adaptive video streams while ensuring better system utilization. *D-VIEWS* is capable of enforcing bit-rate stability by only being aware of the set of video bit-rates, without requiring modifications to the delivery mechanism or other network elements.

Nonetheless, most server-based rate adaptation schemes produce high overhead on the server side with a high complexity, especially when the number of clients increases. Most of these schemes also need modifications to the MPD or a custom server software to implement the rate adaptation logic, which may be perceived as a violation of the DASH standard design principles.

Chapter 3

Evaluation of rate adaptation in mobile networks

3.1 Introduction

Video streaming undoubtedly becomes fast an integral part of the total traffic carried over the Internet, especially in mobile networks [1]. Nonetheless, mobile networks represent a set of challenging network conditions for video streaming as they are often characterized by instability, primarily due to rapid throughput fluctuations. This is attributed to physical phenomena related to radio propagation, such as scattering and path attenuation or handover events, which occur when a data session is transferred to another cell.

Video-on-demand and live streaming services are the most dominant services generating this traffic [2] and to ensure growth, many OTT service providers have invested considerable time and effort to keep pace with ever-increasing demand for robust HAS solutions. While in Chapter 2 we classified HAS solutions according to the module that implements the rate adaptation logic, in this chapter we will focus our analysis particularly on the class of client-side rate adaptation. Client-side solutions are considered the most widespread HAS implementation, since they offer direct control of the rate adaptation process to OTT service providers and require the least assumptions regarding the system architecture. Additionally, they can access directly application level readings, such as the buffer state, without the need of feedback loops with the server. Client-side rate adaptation algorithms are at the core of every major streaming service, and although recent years have seen sustained development of such methods, rate adaptation performance evaluations are scarce. In this chapter we provide a performance evaluation of HAS, in the context of mobile networks.

In particular, we evaluate 5 client-side state-of-the-art rate adaptation methods and we treat every sub-class therein (according to Chapter 2), i.e.

throughput based, buffer-based and hybrid rate adaptation methods. To perform the evaluation we resort to real field-measurements for mobile network traces and we consider two of the most popular streaming applications in terms of traffic generation, i.e. VoD and live streaming [2].

User perceived QoE plays a critical role for the assessment of the various HAS solutions, since it is directly connected to the user engagement and thus, the revenue of content providers. In particular, video stalls and frequent video rate switching are dominating QoE factors for HAS [48]. Avoiding stalls due to the depletion of the client's play-back buffer (smoothness), while at the same time minimizing the frequency of rate adaptation (stability) and providing high average streaming rate, is a very challenging task, especially at high network load or at poor wireless coverage. Moreover, the inherent trade-offs between key video performance measures (i.e average rate, stability and smoothness) makes this attempt even more difficult.

To this end, several client-side rate adaptation algorithms have been proposed, which namely differ in the required input information, according to our analysis in Section 2.1. Throughput-based rate adaptation algorithms, such as *PANDA* [61], rely their decision on the observed throughput, which requires a sufficient number of probes to obtain reliable measurements. In another direction, buffer-based rate adaptation algorithms, such as *BBA* [68] and *BOLA* [69], observe and react to the level of the client's playback buffer. Moreover, hybrid rate adaptation schemes, such as *ABMA+* [79], rely on both the principle of throughput probing and the use of buffer maps, which define the required play-out buffer capacity to satisfy a re-buffering threshold. Despite several research efforts and proposals, there still appears to be a lack of consensus and an ongoing debate regarding the merits within each of above classes of algorithms.

In light of this debate, in this chapter we provide a comparative case study of HAS schemes in mobile networks. Although similar studies, are present in literature, our approach provides novel insight, as it combines for the first time (i) a per class evaluation, (ii) traces form real cellular networks and (ii) different streaming applications.

3.1.1 Prior work

Timmerer et al. [110] make both subjective and objective studies of various throughput-based rate adaptation algorithms, yet, unlike our approach no other class of algorithms is included in this work.

Muller et al. [111] provide a detailed evaluation of the most popular propriety systems, i.e., Microsoft Smooth Steaming [8], Adobe HTTP Dynamic Streaming [9], and Apple HTTP Live Streaming [10]. In particular, these systems are evaluated under the perspective of a vehicular environment.

Akhsahabi et al. [112] focus on an experimental evaluation of two major commercial players (Smooth Streaming [8], Netflix) and an open source player.

Their experimental design covers three important operating conditions: (i) the reaction of the player in short-term changes in the underlying network available throughput (ii) the convergence to the maximum sustainable video rate and the behaviour of multiple players when competing in a bottleneck link and (iii) how does adaptive streaming perform with live content. An investigation of various rate adaptation methods in the context of live streaming can also be found in [113].

Zabrovskiy et al. [114] propose a common evaluation framework for adaptive HTML5 players and demonstrate its applicability by evaluating eight different players which are actually deployed in real-world services.

Our work is more similar to the work of Riiser et al. [115], where the authors compare several adaptive media players on the market to see how they perform in challenging streaming scenarios on a mobile 3G network. They collect throughput data in real-world field trials, and investigate how the media players respond to fluctuating throughput and outages, and how this affects the rate selection process.

We have found that a study that focuses on the categorization of the algorithms according to their input dynamics, along with a performance evaluation on mobile networks that considers multiple streaming scenarios (VoD and live streaming), is missing from the literature. To this end, the work presented in this chapter provides a comparison of the latest rate adaptation algorithms, per class and in the context of mobile networks, which includes buffer-based and hybrid rate adaptation solutions for the first time.

For the interested reader, useful frameworks for automatic evaluation of adaptive video streaming players can be found in [116] and in [117], while an open source multimedia player for testing rate adaptation methods can be accessed in [118].

3.1.2 Contributions

In this work we provide the following contributions:

- We investigate buffer-based and hybrid rate adaptation for the first time and provide insight on the merits of each class of algorithms
- Our comparison is based on the implementation of the algorithms in a single simulation framework, which uses throughput information from publicly available mobile network profiles.
- Furthermore, due to the increasing popularity of live streaming services, which have small buffer sizes due to the strict real-time delay requirements, we also examine the impact of the buffer size on the performance of the rate adaptation algorithms.
- We provide a well specified experimental design that the research community can benefit by replicating in similar studies.
- According to the results of our evaluation, we provide rate adaptation implementation guidelines to designers and operators, for the right

algorithmic approach according to expected network conditions and service requirements.

- Most significantly, we identify that designing robust rate adaptation algorithms for high QoE under changing conditions and requirements, without relying on pre-selected designer-specific parameters or heuristic design, is still a major challenge for multimedia research. Our work presented in Chapter 5 addresses this challenge.

The remainder of the chapter is organized as follows. In Section 3.2 we briefly describe the main principles and properties of the selected state-of-the-art rate adaptation algorithms. In Section 3.3 we describe the validation process followed to obtain our comparison results, including our implementation parameters and simulation factors. Then, in Section 3.4 we present our results on the performance of the rate adaptation algorithms. We conclude with our remarks in Section 3.5.

3.2 Studied rate adaptation algorithms

In this section we briefly present the five state-of-the-art rate adaptation algorithms that were studied, implemented and compared against each other. We have chosen the most representative algorithms, per class, as they have been used in literature for other comparisons.

Throughput-based rate adaptation

Most throughput-based rate adaptation schemes are based on a four-step adaptation model, where initially the available network bandwidth is estimated and then smoothed using noise-filters to avoid estimation errors due to throughput variation. Then, the video rate is indicated based on the discretized output of the smoothing step. The next segment request is scheduled once the inter-request time is estimated.

Conventional [61, Section IV] is a simple rate adaptation algorithm, based on the four step model, which equates the current available bandwidth with the throughput, as it is measured during the previous segment download. Then, the proposed video rate yields by applying an Exponential Weighted Moving Average (EWMA) filter and a dead-zone quantizer. The algorithm determines the inter-request time of the next segment using a bi-modal scheduler, by which the next segment request is scheduled either with a constant delay when the buffer is full or immediately, otherwise.

PANDA [61, Section VI] is an advanced variation of the four-step model above, yet with two distinct modifications. In the estimation step, this algorithm uses a more proactive probing mechanism, that is designed to minimize video rate oscillations. The second modification is at the scheduling step, where a more sophisticated scheduler is considered that drives the

buffer level towards the maximum buffer level B_{max}. At the same time the inter-request time is matched to the necessary time needed to complete the download based on the smoothed estimated value of the available throughput.

Buffer-based rate adaptation

BBA [68, Section VII] is a buffer-based rate adaptation algorithm, proposed by Huang et al. [68], that introduces a segment-buffer map based on the average size of the segments for every encoded rate available at the server. The map is defined by two thresholds: i) an upper threshold that drives the policy to select the maximum encoding rate available, once the instantaneous buffer occupancy surpasses it and ii) a lower threshold that dictates the lowest available encoding rate, if the instantaneous buffer level is lower than that threshold. In the buffer region between these thresholds the policy may use any non decreasing function to select the encoding rate of the next requested segment.

BOLA [69] is a buffer-based rate adaptation algorithm that uses Lyapunov optimization in order to indicate the video rate of each segment. Practically, the algorithm is designed to maximize a joint utility function that rewards an increase in the average streaming rate and penalizes potential re-buffering occurrences. More specifically, a the implemented variation in our simulations called *BOLA-O*, mitigates video rate oscillations by introducing a form of bit-rate capping when switching to higher rates.

Hybrid rate adaptation

ABMA+ [79] is a rate adaptation and buffer management algorithm, which selects the video representation based on the predicted probability of video stalling. The algorithm continuously estimates the segment download time and uses a pre-computed play-out buffer map to select the video rate, which guarantees smooth content play-out. The segment download time estimation is based on the same probing mechanism as *PANDA*, the throughput-based method, but *ABMA+* takes into account VBR aspects as well.

3.3 Experimental framework

3.3.1 Selected network data-sets

The performance of rate adaptation is highly correlated with the network conditions during the streaming. As opposed to fixed networks, mobile networks are characterized by intense throughput variation. Additionally, due to diverse coverage quality there may appear areas with low throughput,

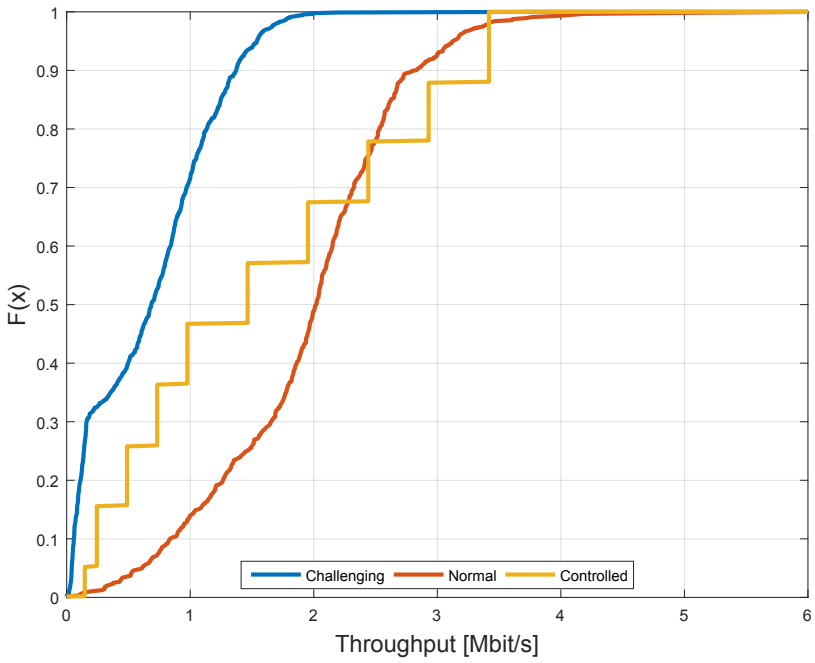


Figure 3.1: Empirical CDF of throughput for the selected network profiles.

which will result to throughput outages and therefore an increased probability of a video stall. In order to avoid prolonged stalls, a rate adaptation algorithm needs, at the very least, a network profile that on average offers a throughput at least higher than the lowest available encoded rate stored at the server. Otherwise, the buffer may be completely depleted.

In order to obtain insightful results for our comparison, we chose two diverse throughput profiles for our simulations, that are representative of a normal and a challenging network profile in vehicular environments. These profiles correspond to direct throughput measurements from a bus and an underground metro respectively [119]. We investigate mobile networks as they are capable of stressing the adaptation methods to highly challenging conditions, as opposed to wire-line networks. In particular, we preferred the use of 3G traces over LTE. Although they are more contemporary and offer higher throughput, LTE networks are currently not always available to all users. Additionally, we have synthesized an artificial profile, which offers controlled network conditions in order to validate our implementations. In Figure 3.1 we can see the Cumulative Distribution Function (CDF) of all studied network profiles.

The *controlled* profile corresponds to a High-Low-High network profile inspired from the experimentation guidelines of the DASH-IF Forum [45]. This profile is shown in Figure 3.3 and it is characterized by the distinct and controlled increases and decreases of the total throughput every 30 s. The

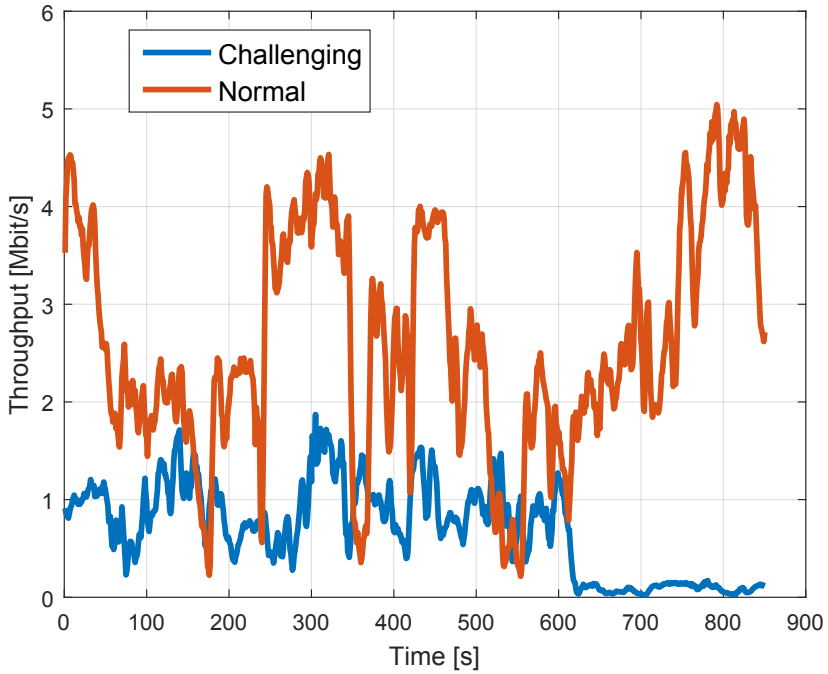


Figure 3.2: Characteristic example of selected real-field network profiles

normal profile is illustrated in Figure 3.2. It is characterized by significant throughput variation, which is expected from real networks. In general, a high throughput is sustained and no significant outages appear. This profile was chosen as representative of a vehicular terrestrial network profile and corresponds to the "bus" data-set as described in [119], consisting of 5 traces; after excluding those that showed long outages. The *challenging* profile corresponds to measurements made on an underground "metro" and consists of a selection of 7 traces which, in general, show a low throughput throughout the route and there is a long outage period when the metro enters a tunnel towards the end of the trace, as depicted in Figure 3.2. This profile allows us to stress the selected rate adaptation algorithms and test their performance under difficult and extreme conditions. We expect to see an increased re-buffering frequency with this "underground" trace-set.

3.3.2 Streaming content

As streaming content, we have chosen 3 representative open movies commonly used for testing video codecs and streaming protocols and recommended in the measurement guidelines of the DASH Industry Forum [45]. The first movie is *Big Buck Bunny (BBB)*, a high motion computer animated movie of 9:56 min duration. The second is *The Swiss Account (TSA)*, which is a

Table 3.1: Video representations

Representation index	Resolution	BBB Max encoding rate (kbps)	TSA Max encoding rate (kbps)	RBPS Max encoding rate (kbps)	CDF Quantiles
1	320×240	129	128	149	0.01
2	480×360	378	330	395	0.05
3	854×480	578	754	700	0.1
4	1280×720	985	1331	1536	0.25
5	1280×720	1536	2048	2048	0.5
6	1920×1080	2353	2764	2560	0.75
7	1920×1080	2969	3481	3072	0.95

sport documentary with regular motion scenes and a duration of 57:34 min. The third is *Red Bull Play Street (RBPS)*, which is a sport show with high motion scenes and a duration of 1:37 hours. For all movies we used the video encodings of [120] in order to obtain the bit-rate levels $r_n \in \{r_1, \ldots, r_N\}$. We selected a total of $N = 7$ encoding rate levels , based on the quantiles of the CDF distribution of the normal network profile (Table 3.1), with a segment duration of $V = 4$ s and a total amount of segments $T = \lceil D/V \rceil$, where D is the length of the played-out video. The particular selection of the representations was made in order to ensure that the minimum rate is sustainable for 99.9% of the normal profile. Of course this would lead to a very small probability of re-buffering for that case, yet it serves as a good basis for the comparison with the challenging profile. Additionally, we chose a high number of distinct representations in order to ensure mild transitions between rate switches. QoE studies [48] suggest that rate adaptation amplitude, is the second most dominant influence factor, after stalling, which means that finer granularity switching may compensate for higher switching frequency. One movie was used per trace, chosen at random to ensure unbiased statistics, and it was repeated if the trace duration was larger than its duration.

3.3.3 Client model and metrics

The client model consists of the maximum buffer level B_{max} and the selected rate adaptation algorithm that the player may deploy during a video streaming session that makes a quality representation decision $x_t \in \{1, \ldots, N\}$ for every segment $t \in \{1, \ldots, T\}$, that yields an encoding rate $r_{x_t} \in \{r1, \ldots, r_N\}$. We ran our experiments over 12 mobile traces (7 normal network traces and 5 challenging network traces). Furthermore, we investigated the maximum buffer occupancy factor B_{max}, since various applications (live, VoD, short clips or long movies) may target different maximum buffer occupancy levels. In particular we repeated the experiment for a small $B_{max} = 16$ s (4 segments), which simulates a live streaming service and for a larger $B_{max} = 92$ s (23 segments), to simulate the case of VoD. These studied values were selected based on measurements of the maximum buffer level of a popular streaming service, which offers both live streaming and VoD streaming. We assert that

our results hold for any buffer value larger than 4 segments, but leave the full study of the impact of the buffer level to future work. Also two important parameters that may affect the QoE of the user are the initial buffering (i.e the amount of segments that need to be downloaded in the buffer before play-out can start initially) and the re-buffering threshold (i.e the amount of segments that need to be downloaded in the buffer before play-out can resume after a stall event). These parameters were both set equal to $\tau = 2$ segments for our experiments, as indicated in the implementation guidelines of the studied algorithms.

Since a consensus on a unified framework for measuring QoE is missing from the scientific community, in this work we resort to quantitative performance metrics that express the main influence factors of video streaming QoE; namely average (video) rate, stability, smoothness, continuity and consistency. Average video rate measures to what degree the rate adaptation algorithm is able to adapt to the average throughput experienced during a segment download, whilst stability and smoothness are measures of how often and at what degree (respectively) the rate adaptation algorithm switches to a new bit-rate level. Continuity and consistency are related to measuring the effects of session interruptions (stalls), in terms of frequency and duration, respectively. Both are indicators of the ability of the rate adaptation algorithm to provide uninterrupted streaming. Following we provide a formal definition for these performance metrics.

Average rate models the average video bit-rate $\bar{r} = \frac{\sum_{t=1}^{T} r_{x_t}}{T}$ of the received video segments in a session, normalized over the maximum average rate obtained $\max_{m \in \mathcal{M}} \bar{r}^m$ for that session by any algorithm $m \in \mathcal{M}$, where $\mathcal{M}$ is the set of all evaluated algorithms. It is calculated with the term:

$$\frac{\bar{r}}{\max_{m \in \mathcal{M}} \bar{r}^m}. \tag{3.1}$$

Stability and smoothness of a streaming session are associated with the rate adaptation frequency and the amplitude of rate adaptation, respectively. Adaptation frequency (or stability) is the number of rate switches over the total number of segments that allow a rate switch $T - 1$, given by the term:

$$\frac{\sum_{t=2}^{T} (\mathbb{1}_{\{r_{x_t} \neq r_{x_{t-1}}\}})}{T - 1}, \tag{3.2}$$

where $\mathbb{1}_{\{\mathcal{Y}\}}$ is an indicator vector; with ones at the positions that condition $\mathcal{Y}$ is true, and zeros otherwise. Adaptation amplitude (or smoothness) is the normalized average distance per rate switch, in terms of bit-rate. It is calculated with the term:

$$\frac{\sum_{t=2}^{T} |r_{x_{t+1}} - r_{x_t}|}{r_N \sum_{t=2}^{T} (\mathbb{1}_{\{r_{x_t} \neq r_{x_{t-1}}\}})}. \tag{3.3}$$

When considering consistency of a session we must take into account the re-buffering duration, while when measuring continuity of a streaming session we look at the frequency of re-buffering events. Re-buffering duration (or consistency) is the total duration of re-buffering events in a stream over the length of the played-out video D:

$$\frac{\sum_{t=\tau+1}^{T} \rho_i \cdot (\mathcal{T}_t^{end} - \mathcal{T}_t^{start})}{D}, \tag{3.4}$$

where $\rho_i = 1$ if a re-buffering event occurred during the download of segment i and $\rho = 0$ otherwise and $\mathcal{T}_t^{end}$ and $\mathcal{T}_t^{start}$ are the time of end and the start of the re-buffering event, which occurred during the download of segment t, respectively. Re-buffering frequency (or continuity) is the number of re-buffering events that occurred in a stream over the number of segments T, measured by:

$$\frac{\sum_{t=\tau+1}^{T} \rho_t}{T}. \tag{3.5}$$

In the figures of Section 3.4, these metrics are averaged over the complete set of traces for every profile and we present the standard error with a confidence level of 95%.

3.4 Results

In this section we evaluate the performance of each rate adaptation algorithm based on the metrics introduced in Section 3.3. The results are not standalone and a combination of the performance metrics is required for the evaluation of the algorithms. The scope of this work is not to introduce a QoE model but to present the raw results of the selected metrics.

Implementation validation

In Figure 3.3, the throughput profile (controlled) that was used to validate our implementations, is shown as an indication of the behavior per class of algorithms. We can see as a first difference, that buffer-based rate adaptation starts with a low rate and gradually increases it while the buffer fills up given that the throughput is higher than the streaming rate. Contrary, throughput-based methods estimate the available throughput, through probes, and match it immediately to the available throughput. The hybrid rate adaptation starts with a low rate until the algorithm has a sufficient amount of probes to estimate the download time appropriately. The second significant difference is that buffer-based and hybrid rate adaptation may select a rate higher than the available throughput for a short period as potential throughput drops have not, yet, decreased the buffer level beyond a threshold or registered in the time-probing samples, respectively. Alternatively, the throughput-based

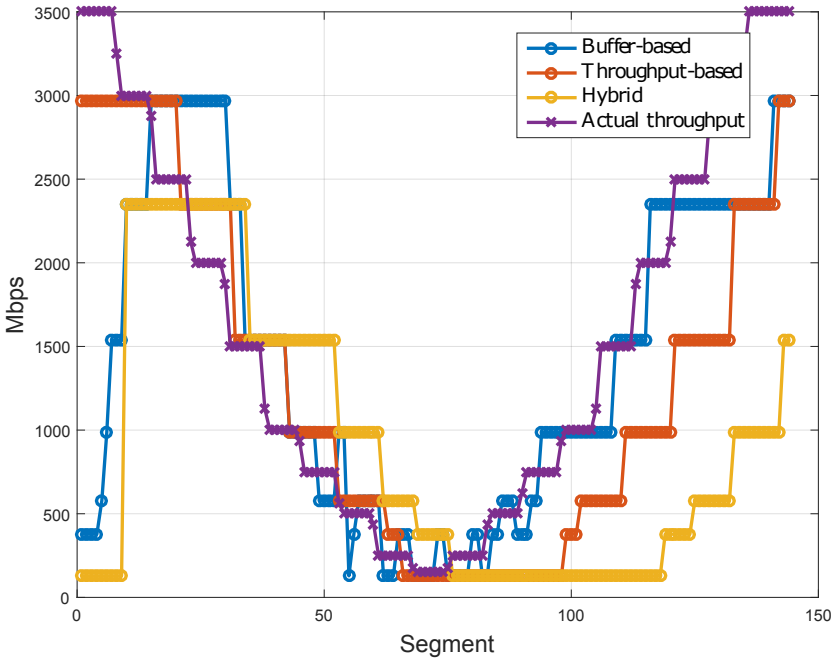


Figure 3.3: Per class rate selection example for a controlled network profile

algorithms can be more reactive. It is evident that hybrid rate adaptation has a small delay in the rate adaptation to the current throughput, as the throughput variation is registered in the time-probing sample as an average of the last 50 probes (sampling window [79]). At this moment it is worth mentioning that all studied algorithms were designed either heuristically or based on pre-selected parameters. In all our implementations we used the parameterization proposed by the designers, but we note that better results could be achieved if the parameters were fine-tuned to the considered network and application scenarios.

Average rate

Figure 3.4 shows the results for every algorithm individually and in parallel we report the results in average rate for both network profiles ('Challenging' and 'Normal') and for the cases of VoD and live streaming (indication of the term 'Live'). From this figure we report that buffer-based algorithms achieve the highest average rate in all conditions. They are more successful, by design, in conserving high bit-rate levels, regardless of potential throughput fluctuation as the decisions typically look at the buffer state, which exists precisely to absorb such fluctuations. Throughput-based and hybrid algorithms show a slightly diminished ability to match the rate

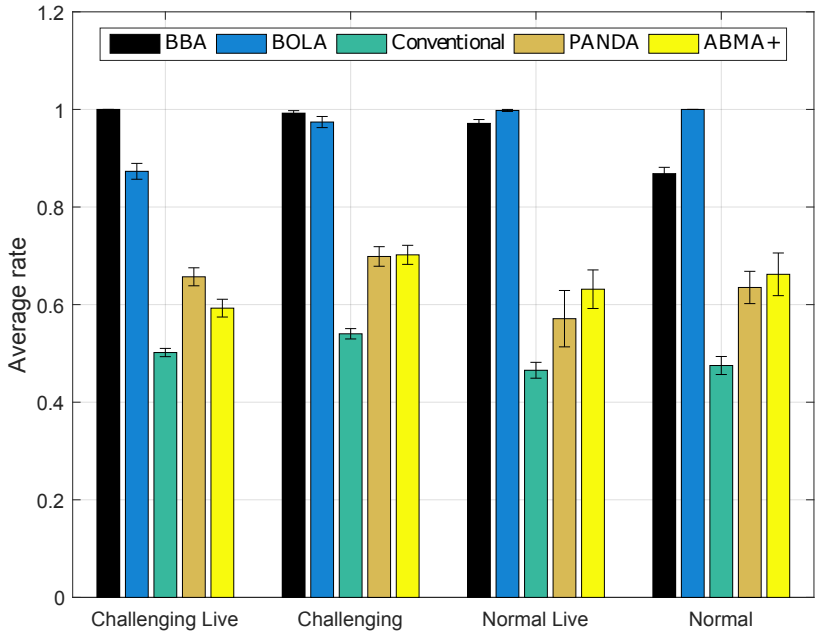


Figure 3.4: Average rate

to the available average throughput due to the throughput variation that characterizes the selected network profiles and that they are designed to estimate. We also notice from this figure that the buffer size does not affect significantly the average rate.

Consistency and continuity

As far as frequency of re-buffering events (continuity) is concerned, Figure 3.5 shows that, as expected, the probability of a re-buffering is slightly higher in the cases of a small buffer (i.e live streaming). A small buffer has limited resilience to throughput variation. Regarding the re-buffering frequencies per class of algorithms, we can note that although the performances are very close, on challenging profiles buffer-based algorithms, along with *PANDA*, are slightly more probable to experience a re-buffering event. This, combined with the higher rates achieved in Figure 3.4 hinds that the smaller buffers associated with live streaming, make buffer-based rate adaptation methods have a more aggressive adaptation behavior. For normal scenarios we witness continuous streaming from almost all algorithms, due to the absence of long throughput outages in this profile and also hinds a well designed simulation framework (selection of rate levels based on quantiles of normal profile). Nevertheless, *ABMA+* shows a slightly decreased continuity, when

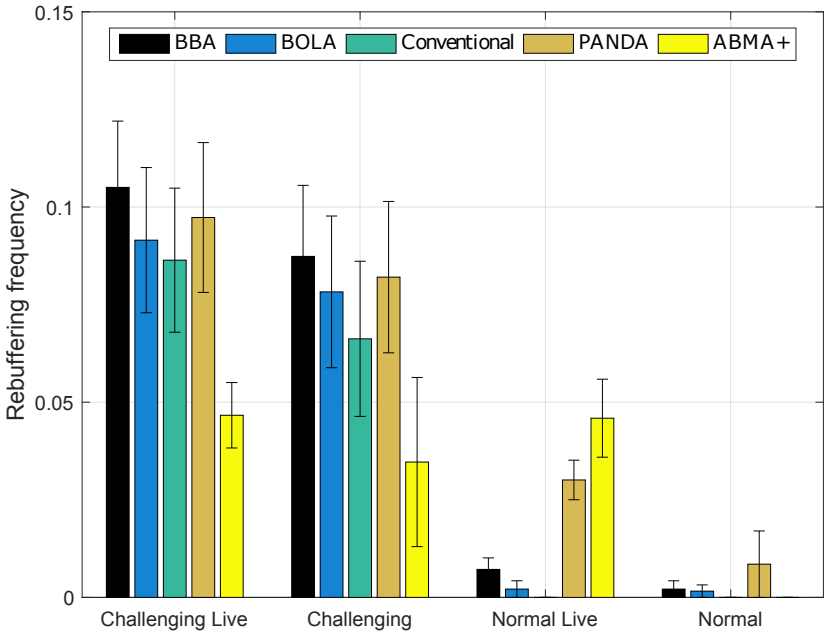


Figure 3.5: Re-buffering frequency - continuity

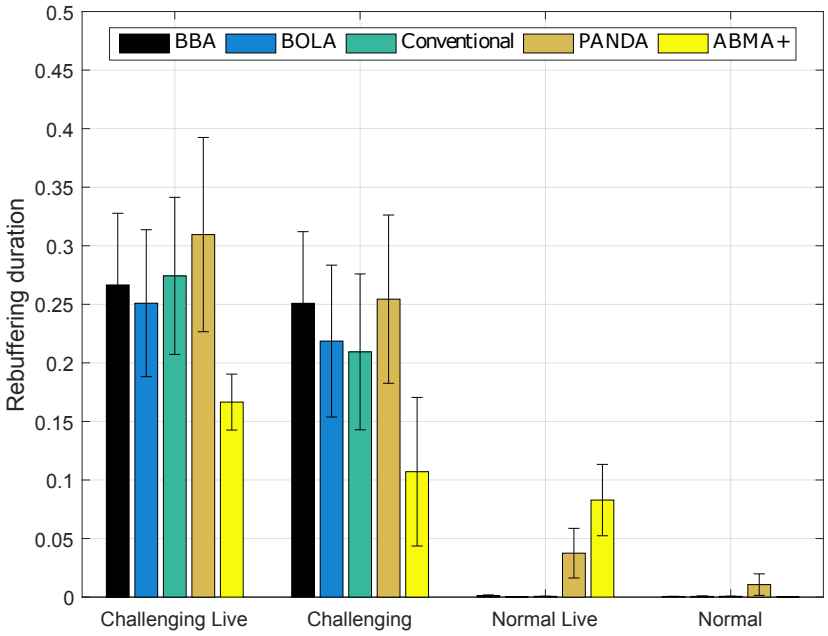


Figure 3.6: Re-buffering duration - consistency

compared with the rest of the algorithms in the normal live profile, probably due to the fact that the number of probes (50) proposed by the algorithm designers is very high compared to the maximum buffer size. Therefore a short throughput drop is not registered in the estimation in appropriate time before the buffer has been depleted. Figure 3.6 shows the duration (amplitude) of the re-buffering events (consistency), where we see re-buffering events lasting about 25% of the video duration. This is expected since the challenging profile includes underground areas, which may cause network outages, for 1/4 of the trace.

Stability and smoothness

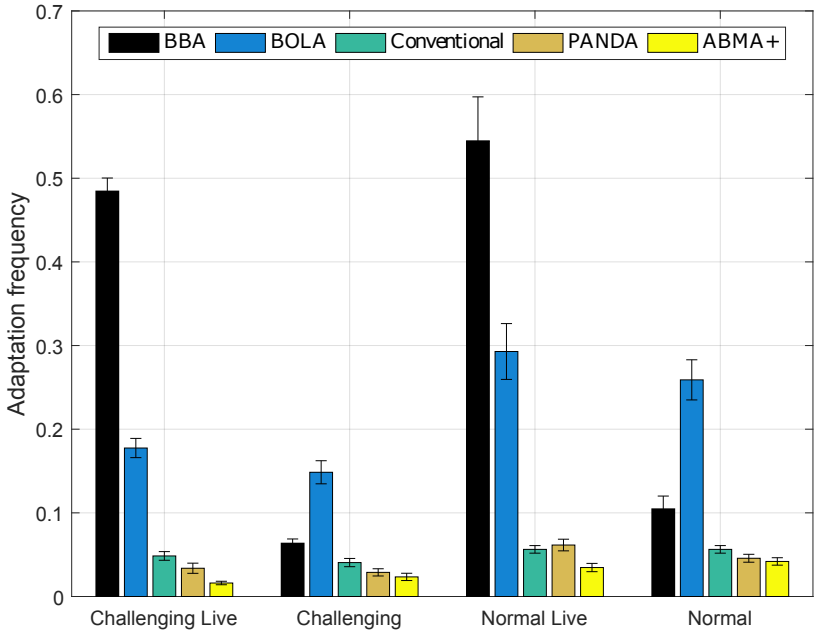


Figure 3.7: Rate adaptation frequency - stability

Last but not least, rate adaptation frequency (stability) has a very significant impact on perceived streaming QoE [48]. Figure 3.7 shows that buffer-based algorithms are about 40% more probable of making a rate switch when the buffer is small. *BOLA* is optimized to achieve a high efficiency but the stability aspect is not considered in the optimization, since it is addressed with a heuristic in a second phase. *BBA* has a pre-selected constant higher buffer threshold which makes the segment map less agile to throughput variation when the maximum buffer is small. On the contrary, throughput-based and hybrid algorithms appear to switch rate less often, but according

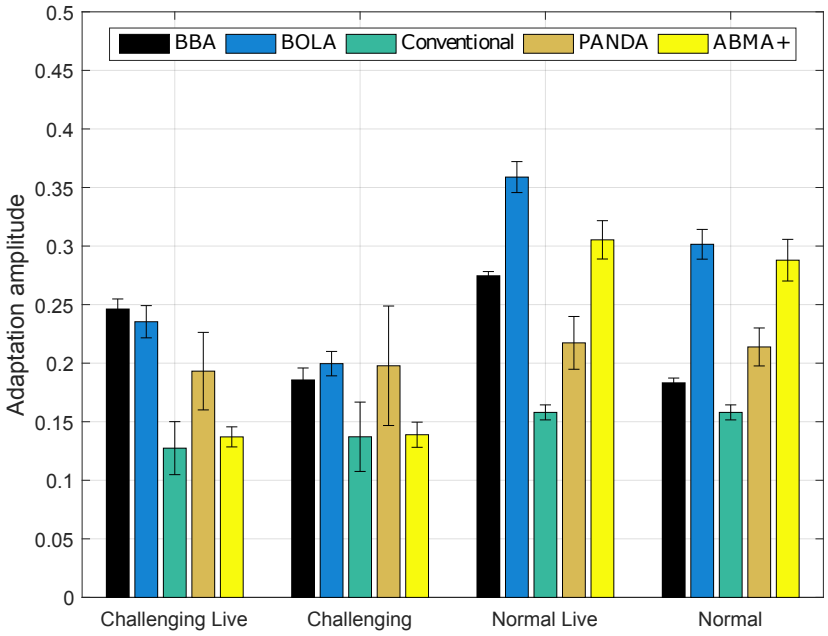


Figure 3.8: Rate adaptation amplitude - smoothness

to Figure 3.8, with similar rate adaptation amplitude (smoothness), which is complementary to rate adaptation frequency (stability). Figure 3.8 shows the average normalized distance between rate switches (smoothness). The performance regarding this metric shows a slight advantage in favor of the throughput-based algorithms, in both normal and challenging profiles.

Table 3.2: Best client-side rate adaptation class per QoE metric

Buffer Size	**Network condition**	**Average rate**	**Re-buffering frequency**	**Re-buffering duration**	**Adaptation frequency**	**Adaptation amplitude**
Small (16 s)	Normal	Buffer	Throughput	Buffer	Hybrid	Throughput
Small (16 s)	Challenging	Buffer	Hybrid	Hybrid	Hybrid	Throughput
Large (92 s)	Normal	Buffer	Hybrid	Buffer	Hybrid	Throughput
Large (92 s)	Challenging	Buffer	Hybrid	Hybrid	Hybrid	Throughput

It is important to mention that no metric should be treated separately and that only the combination of all metrics allows our comparison to be insightful. Overall, our results match those in [69] with the addition that *BOLA* is compared against another buffer-based rate adaptation for the first time. Moreover, our results can be verified with those in [79] where *PANDA* and *BBA* are compared against *ABMA+*. In Table 3.2 we have gathered the best performing classes of algorithms, per QoE element. This table can serve as insight to the selection of the most appropriate algorithmic class, depending on the application parameters (live streaming, VOD, etc.) and

the commonly experienced network conditions.

3.5 Conclusion

In this study, we evaluated the performance of five state-of-the-art rate adaptation algorithms and made a per class comparison, based on network traces for two different throughput profiles. Additionally, we evaluated the maximum buffer factor to see how each strategy behaves for smaller and larger buffers, as different services may target different buffer levels. Our conclusion is that buffer-based approaches outperform any other class of algorithms in terms of average rate, yet they may lack in stability, especially for small buffers, common in live streaming services. Based on these results, in Chapter 4 we extend a common buffer-based adaptation method by a Bayesian coverage estimator, incorporating inferred location information into the rate adaptation process. Thus, we investigate the merits of context-aware rate adaptation, especially in buffer-based adaptation, leveraging modern-day mobile device capabilities in accessing cross-layer information.

The work presented in this chapter provides first guidelines to designers and operators of rate adaptation algorithms for the right algorithmic approach according to expected network conditions and service requirements. Additionally, a significant insight gained from the performance evaluation, presented in this chapter, regards the identification of a fundamental limitation in the design of rate adaptation methods. Relying on pre-selected designer-specific parameters or heuristic design, constitutes still a major challenge for multimedia research, when designing robust rate adaptation algorithms aiming at high QoE under changing conditions and requirements. Therefore, in a direction to mitigate this challenge, in Chapter 5 we recast rate adaptation under an optimization framework and present a novel rate adaptation algorithm based on online learning, that does not require any statistical assumptions for the unknowns.

Chapter 4

Context-aware adaptive streaming

4.1 Introduction

The constant evolution of mobile networks, mainly 4G's LTE, along with recent progress in the capabilities and affordability of personal mobile devices, such as modern day smartphones, have surged the demand for mobile multimedia consumption, which now constitutes a significant fraction of all global IP-traffic [1]. Nonetheless, HAS, the main multimedia delivery technology, often fails in mobile networks, that are commonly characterized by an overall fluctuating QoS and in particular by throughput variation. Depending on the current user location, mobile throughput may change drastically as a result of physical effects during radio propagation. While the client's playback buffer typically protects the stream from short channel outages caused by fast fading, slow effects such as path loss and shadowing can cause much longer throughput decreases. Particularly users may lose coverage when entering a building due to the high signal attenuation at windows and walls. Being unaware of the state of the wireless channel, most rate adaptation policies may react too late and too conservatively, when compensating for long wireless throughput degradation. This is a common case if a user is obstructed by a large obstacle or moves inside a tunnel or a building [121].

The incorporation of context information, such as user location, into the rate adaptation process gives rise to a new paradigm in HAS, namely that of *context-aware rate adaptation*. Typically rate adaptation algorithms would look at previous throughput measurements in order to estimate the available throughput at the client, and based on these estimates the rate adaptation algorithm would select the appropriate streaming rate. Often additional information might be considered, such as application-level readings; namely the instantaneous buffer occupancy. As smartphone technology is constantly complemented with more and more sensors, modern mobile devices presently

have access to a wider spectrum of information coming from across the OSI stack [12]. In this work we propose a method that leverages such cross-layer information to enhance mobile HAS.

In this chapter we study how context-awareness can be combined with the adaptive streaming logic to design a proactive client-based video streaming strategy. Our proposed algorithm, called *Indoors-Outdoors Buffer-Based Adaptation (IOBBA)* aims to provide a fast and aggressive buffer-based rate adaptation decision if (and only if) the user's current mobile coverage requires it. Thus, in this chapter, we present a Bayesian detector that combines measurements from two smartphone sensors to decide if a user is inside a building (indoors) or not (outdoors). Based on this coverage classification, we then propose a rate adaptation algorithm to increase playback stability, while maintaining a high average rate. Measurements in a typical office building show high accuracy for the presented detector and improved QoE for the proposed rate adaptation algorithm.

4.1.1 Prior work

Similar to our approach, Liotou et al. in [121] focus on how context-awareness can enable more strategic decisions towards guaranteeing a seamless viewing experience of the end-users. The authors use information about the users direction, speed and trajectory combined with a terrain map to predict whether the user will pass through a tunnel, in which case a more conservative HAS policy is employed aiming to boost the buffer level by downloading lower qualities for a period of time.

Triki et al. [122] propose a scheduling mechanism for video streaming traffic, in which the access to the network resources is restricted to users with a Signal-to-Interference-plus-Noise Ratio (SINR) above a given threshold. This scheme benefits from the fact that, as users are in permanent motion, they may experience different SINR values during the same video streaming session offering the opportunity to only schedule users with good channel conditions. The proposed context-aware scheduling approach not only allows to achieve overall network spectral efficiency improvement, but also guarantees fairness and QoE among users.

Particularly for mobile clients that are in motion, the network conditions are more fluctuating with respect to spatiotemporal factors. Studies, such as that of Riiser et al. [115] or Yao et al. [123], deploy a bandwidth lookup service to guide the throughput estimation among the mobile clients. However, most of these methods regard the fluctuation of throughput primarily under the location variation rather than its evolution with time.

Since it takes a long time until a coverage loss affects application-layer buffers and throughput estimators, the mobile streaming community started to include information from lower protocol layers into rate adaptation policies. In LTE networks, *piStream* [124] by Xie et al., enables clients to estimate

the available throughput based on physical-layer measurements.

While *piStream* obtains the link-layer throughput directly from the LTE scheduling grant, Bao et al. [82] decided to adapt to the received signal power. Although it cannot be directly translated into throughput, signal power provides an accurate measurement of the current coverage situation and is readily available at the application layer of modern smartphones without additional battery consumption.

Following the idea of characterizing a user's coverage situation by signal power, we go a step further. In addition to the signal power received from the cellular modem, we use information from the smartphone's localization sensors, i.e. its Global Navigation Satellite System (GNSS) receiver, to indicate a coverage loss. In Section 4.5 we provide strong experimental evidence that this combination highly increases the estimation accuracy compared to using cellular signal power alone.

4.1.2 Contributions

Briefly, the contributions of this work are the following:

- *Indoors-outdoors detector:* In Section 4.3, we present a Bayesian detector that combines signal measurements from the cellular and GNSS receiver. The result is a binary estimate (i.e., indoors or outdoors) of the user's coverage situation in real time.
- *Indoors-outdoors-aware rate adaptation algorithm:* In Section 4.4, we apply the indoors-outdoors detector described in Section 4.3 to improve the performance and stability of a classic rate adaptation method [68].
- *Experimental verification:* In Section 4.5, we present the setup and results of our measurement campaign.[1]. The results consistently show substantial gains in mitigating stalls and stability, compared to the baseline method. [68].

These results demonstrate that accounting for the coverage *context* of a user is a promising approach for designing adaptive streaming policies.

4.2 System model

According to Shannon's channel capacity theorem [125], the maximum attainable error-free data rate is computed based on measurements of the received signal power as follows. Achievable rate at time index $t \in \{1, \ldots, T\}$ is given by:

$$C_t = b_t \log_2(1 + \phi_t), \tag{4.1}$$

where ϕ_t is the SINR and b_t is the user effective bandwidth which corresponds to 90% of the total bandwidth in the 4G standard [126]. We assume that

[1] A video illustrating the measurement scenario and the QoE gain for adaptive streaming is available at `https://youtu.be/qJibE2N37Yk`.

$\phi_t = P_t/\ell_t$ where P_t is the received signal power measured over bandwidth b_s and ℓ_t is the sensitivity threshold, above which the receiver is capable of detecting and processing the received signal. We truncate the value of ϕ_t at 20dB since the highest modulation scheme is obtained at SINR >= 20dB.

The received signal power is available at the application layer of most mobile devices [127]. Without loss of generality, we assume that interference is included in the sensitivity threshold. Since the studied mobile devices may associate to a different mobile network type any time, thresholds for 2G, 3G and 4G have to be considered. We summarize all used parameters in Table 4.1. Details are provided in the standards [128–130] for 2G, 3G and 4G, respectively.

Parameters	2G	3G	4G
Received power P_t	RSSI	RSCP	RSRP
Measured bandwidth b_s	200kHz	3.84MHz	15kHz
Data bandwidth b_t	200kHz	5MHz	18MHz
Sensitivity threshold ℓ_t [dBm/b_s]	-104	-106	-94

Table 4.1: Parameters for the studied mobile networks

Using the obtained SINR, the cell throughput is then computed over the complete bandwidth of the respective network type as in (4.1). Computing this quantity for the complete data set provides the empirical CDF in Figure 4.1.

To obtain the throughput for a single user, we assume a round-robin scheduling policy where all users are served equally. Thus, the total cell throughput is equally shared among K users as $C_{t,k} = C_t/K$ where $k \in \{1, \ldots, K\}$. Note that we have to limit our study to $K \leq 8$, since this is the maximum number of users per channel in 2G [131].

The received signal power P_t was measured over several days inside and outside of a typical multi-floor office building in downtown Paris, France. Off-the-shelf mobile devices were used, namely five HUAWEI Mate 8 smartphones running Android 6.0 Marshmallow [132] with different network providers. The smartphones automatically selected all transmission (e.g., Physical-layer rate) and associated to 2G, 3G, or 4G networks over the course of the measurements. All data was recorded at the application of these devices without direct access to the mobile networks. In addition to signal power, localization data and ground truth about the user's position (indoors or outdoors) were recorded.

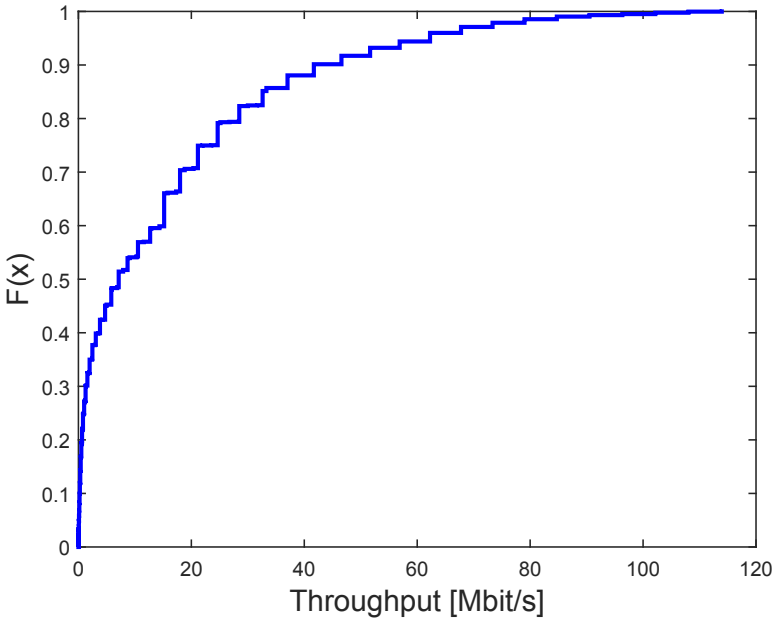


Figure 4.1: Empirical CDF of the cell throughput in the down-link

4.3 Indoors-outdoors detection

Here, we present our Bayesian classifier for a user's coverage situation. Our algorithm detects if a user is indoors or outdoors, based on the a posteriori information from smartphone sensors. The first input is the received signal power as provided by the cellular modem according to [133]. The second input is the confidence radius of the location, provided by the active localization sensor, e.g., the smartphone's global positioning system (GPS) chip set. In the following, we will discuss the statistics of both input variables and describe a method to combine them for indoors-outdoors detection.

4.3.1 Analysis of received signal power

Figure 4.2 shows the empirical CDF of the received signal power measurement, separated by the recorded ground truth (i.e., indoors or outdoors). The significant offset between the two CDFs results from substantial differences in radio propagation, namely reflection, diffraction, and attenuation [134]. Based on these statistics, we conclude that received signal power can be used to classify the user's coverage situation (i.e., either indoors or outdoors). We will proceed using the empirical probability density functions (pdf) of the received power, obtained by differentiating previously recorded CDFs.

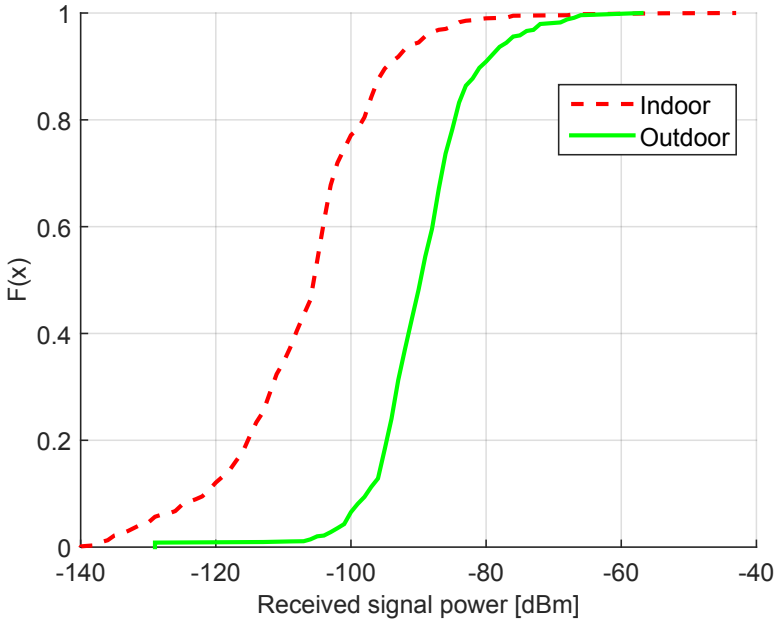


Figure 4.2: Empirical CDF for measured received signal power

4.3.2 Analysis of confidence radius

To improve the detection performance, we propose the use of the confidence radius as provided by the localization system of mobile devices, e.g., [135]. It is important to highlight that confidence radius is the sole localization parameter used by our algorithm. Other parameters such as user position, speed and altitude are not required.

The empirical analysis of the confidence radius shows sparse data for indoors and outdoors measurements as in Figure 4.3. Such quantization is common in GPS chip sets and can be overcome by curve fitting, as shown by the dashed lines. The resulting curves will be used in the following algorithm.

4.3.3 Data fusion and detection

The indoors-outdoors detection is based on maximum *a posteriori* (MAP) estimation, where the unknown *status* maximizes the a posteriori probability under a given observation. The input from the two sensors is combined by computing a joint a posteriori probability based on the distributions of (i) the received signal power and (ii) the confidence radius.

We denote by y_t and z_t the received signal power measure and the confidence radius value, respectively, at time index t. The a posteriori probability is given by $APP(L_t) = P(L_t|y_t, z_t)$, where $L_t \in \{Indoor, Outdoor\}$. Following Bayes' law, the a posteriori probability becomes

$$APP(L_t) = \frac{P(L_t, y_t, z_t)}{P(y_t, z_t)} = \frac{P(y_t, z_t|L_t)P(L_t)}{P(y_t, z_t)}. \tag{4.2}$$

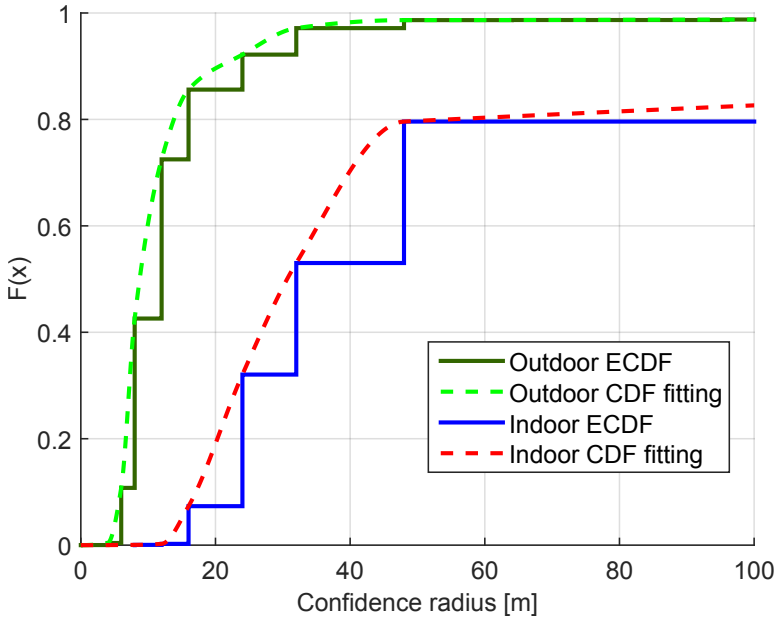


Figure 4.3: Empirical CDF for measured confidence radius

Unlike for received power and confidence radius, no a-priori information is available on L_t and the two input variables are obtained from physically independent sensors. Consequently, the a posteriori probability is proportional to the product

$$APP(L_t) \propto P(y_t, z_t|L_t) = P(y_t|L_t)P(z_t|L_t) \tag{4.3}$$

of the conditioned measurements. Here, $P(y_t|L_t)$ is the direct observation based on received signal power and $P(z_t|L_t)$ is the direct observation based on the confidence radius. Finally, the a posteriori probability can be expressed simply as

$$APP(L_t) \propto obs_y(L_t)\ obs_z(L_t), \tag{4.4}$$

where $obs_y(L_t) = P(y_t|L_t) = \frac{p(y_t|L_t)}{\sum_{L_t} p(y_t|L_t)}$ and $obs_z(L_t) = P(z_t|L_t) = \frac{p(z_t|L_t)}{\sum_{L_t} p(z_t|L_t)}$ are the observations computed from received power and location measurements, as described previously.

The coverage classification (i.e., indoors or outdoors) is then given by the most probable value of $APP(L_t)$, i.e., the value maximizing the a posteriori probability.

4.4 Context-aware rate adaptation

In this section we will describe a rate adaptation policy based on coverage classification.

4.4.1 Baseline buffer-based rate adaptation

A classic buffer-based rate adaptation algorithm is described in [68, Section VII]. This algorithm employs a segment map which is defined in the space $[0, B_{max}]$ and $[\bar{S}_{r_1}, \bar{S}_{r_N}]$, where $\bar{S}_{r_i}$ is the mean segment size of encoding rate level r_i, $i \in \{1, 2 \dots N\}$, as depicted by the black line of Figure 4.4. The map is defined by two thresholds: i) an upper threshold θ_2 that drives the algorithm to select the maximum available rate (r_N) once the instantaneous buffer occupancy $B_t > \theta_2$, for segment $t \in \{1, \dots .T\}$ ii) a lower threshold θ_1 that dictates the lowest available rate (r_1) if $B_t < \theta_1$. In the buffer region $\theta_1 < B_t < \theta_2$, the algorithm may use any non-decreasing function to select the available bit-rate of the next segment to be downloaded [68]. We will refer to this rate adaptation principle, where the non-decreasing function is linear (black line in Figure 4.4), as the *baseline* for the remainder of this chapter.

4.4.2 Indoors-Outdoors aware buffer-based rate adaptation

Let us now present *Indoors-Outdoors aware Buffer Based Adaptation (IOBBA)*, a rate adaptation algorithm that modifies the baseline [68, Section VII], by incorporating coverage classification. *IOBBA* aims at minimizing the re-buffering events for mobile streaming users that move into an area with poor coverage, e.g., inside a building. To this end, the rate adaptation algorithm has to react faster to throughput decreases in order to avoid buffer under-runs. Moreover, the algorithm should increase the video rate conservatively, in order to avoid video rate oscillations at the edge of a poorly covered region. Both design principles should add a minimum penalty to the average video rate. Following these targets, we made two modifications to the baseline algorithm [68, Section VII].

Instead of using a linear function for the buffer region $\theta_1 < B_t < \theta_2$, we employ the function $f(B_t) = \lambda_1 \cdot \lambda_2^{B_t}$, where λ_1 and λ_2 can be computed at the boundary values $f(\theta_1) = \bar{S}_{r_1}$ and $f(\theta_2) = \bar{S}_{r_N}$, respectively. This exponential function is illustrated by the red line in Figure 4.4. To increase stability and to reduce the probability of buffer starvation, we use a constant value for θ_1. This allows for a fast reaction to the immediate buffer depletion, which is a direct consequence of the throughput reduction, when moving into a building. This updated segment map is used when the user location is detected indoors. In this way, the user will experience smoother buffer depletion and thus a reduced probability of re-buffering events. To prevent further loss in the mean video bit-rate, a linear function is used instead, as soon as the user's location is detected outdoors.

Indoors, the *IOBBA* algorithm increases video rate more conservatively than outdoors. While the user is considered to be indoors, video rate is only increased after m sequential requests upgrade requests have been indicated

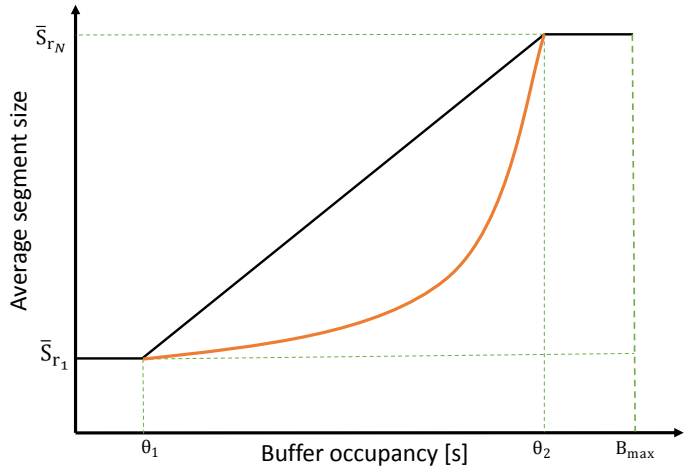


Figure 4.4: Segment maps for the baseline (black line) and the proposed IOBBA algorithm (red line)

by the segment map. With this principle, video rate oscillation becomes less likely.

4.5 Measurement results

We experimentally evaluate various technical influence factors that affect QoE for IOBBA. For each factor, we study the difference to ground truth and to a baseline algorithm from literature.

4.5.1 Methodology

To study the effect of estimation error on the QoE, we study *IOBBA-True* where our rate adaptation algorithm from Section 4.4 operates on the ground truth and *IOBBA-Detected* where the result of our MAP estimator is used. During our one-week measurement campaign, we obtained 35 traces files for a variable number of mobile users ($k \in \{1, 2, \ldots, 8\}$). Furthermore, we investigated the maximum buffer occupancy B_{max}. This factor is important as it may substantially differ between different streaming services (e.g., tens of seconds for live streaming, hundreds of seconds for video-on-demand).

To parameterize our rate adaptation algorithm, we computed the lower threshold dynamically as in [68], for the case that the user is outdoors. For the case that the user is indoors, we used the constant value $\theta_1 = 0.3 \cdot B_{max}$, as indicated [68]. The upper threshold is constant with $\theta_2 = 0.9 \cdot B_{max}$ for both cases (indoors and outdoors) and both algorithms. The video rate increase threshold is $m = 3$ for *IOBBA* when a user is indoors, while $m = 1$ for the baseline and *IOBBA* when a user is outdoors.

Table 4.2: Video content characteristics

Quality index	Resolution	Max encoding rate (kbps)	Quantiles of throughput distribution
1	320×240	129	0.1
2	480×360	378	0.2
3	854×480	578	0.25
4	1280×720	1536	0.4
5	1920×1080	3993	0.5

As streaming content, we have chosen the open movie Big Buck Bunny (BBB) [136], which is recommended in the measurement guidelines of the DASH Industry Forum [45]. BBB is of 9:56 minutes duration and high motion. We encoded the movie with H.264 at 24 frames per second, using MP4 containers at a segment duration of 4 seconds. Following the recommendations of [120], we selected the video encoding rate levels according to the quantiles of the total throughput CDF, as in Table 4.2.

As QoE factors we studied:

1. *Average rate* over the video duration.
2. *Re-buffering events*, which is the number of re-buffering events in the streaming session, typically associated with streaming *smoothness*.
3. *Adaptation events*, which accounts for the amount of video rate changes in the streaming session, typically associated with streaming *stability*.

For each metric, we present arithmetic means over all measurements and the standard error at a confidence level of 95%.

4.5.2 Accuracy of indoors-outdoors detection

The detection is performed by the MAP estimator defined in Section 4.3, based on the pdfs from previous measurements.

The confusion matrix in Table 4.3 shows the accuracy of this estimation when only the received power is used as an input.

Table 4.3: Confusion matrix for indoors-outdoors detection based on received power

Detected(D) \ True state (S)	Indoor	Outdoor
Indoor	**0.8256**	0.083
Outdoor	0.1744	**0.917**

Including the confidence radius as a second input leads to the detection accuracy in Table 4.4.

Table 4.4: Confusion matrix for indoors-outdoors detection based on data fusion

Detected(D) \ True state (S)	Indoor	Outdoor
Indoor	**0.9496**	0.0911
Outdoor	0.0504	**0.9089**

This clearly shows the benefit of including this second variable as a 12% accuracy increase for the indoor case. Over all cases, the MAP estimation over both variables reaches an accuracy of $P_{io} = 92.98\%$.

4.5.3 Performance evaluation

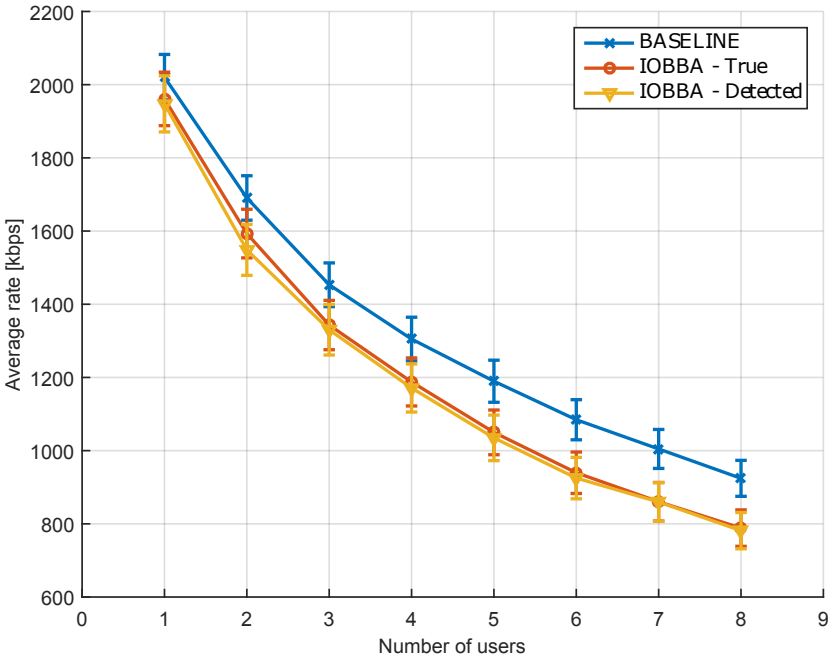


Figure 4.5: Average rate vs. users for $B_{max} = 150s$

Figure 4.5 to Figure 4.7 show the selected performance indicators for an increasing number of concurrent channel users $k \in \{1, \ldots, 8\}$, while the maximum buffer level is kept constant at $B_{max} = 150s$.

As the number of users increases, the bandwidth-share assigned to each user is reduced and therefore the average video rate is decreased, as depicted in Figure 4.5. Additionally, Figure 4.7 shows that the re-buffering events are experienced more frequently as the number of users increases, since the lower the throughput of each user the higher the probability of a buffer

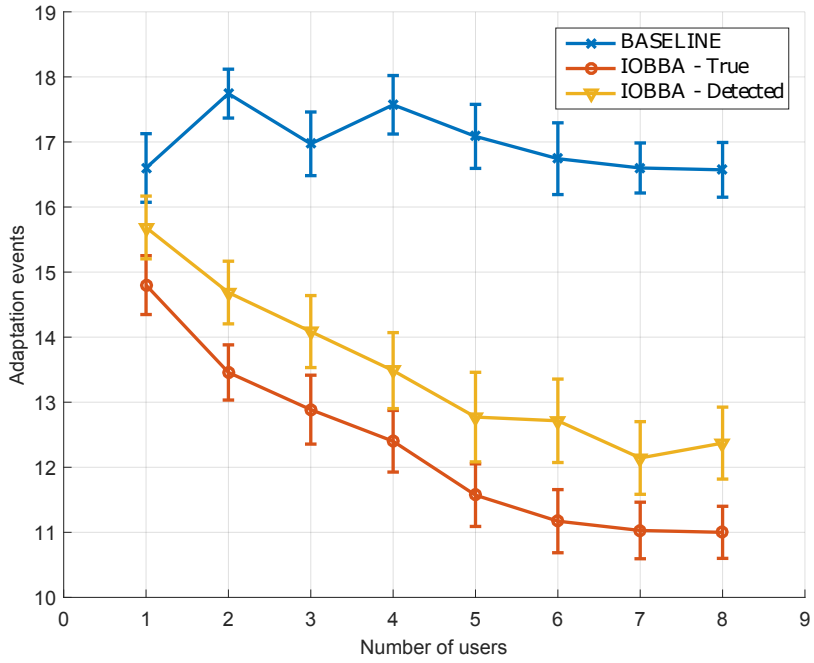


Figure 4.6: Adaptation events vs. users for $B_{max} = 150s$

under-run. For the rate adaptation events, in Figure 4.6 we notice that using indoor-outdoor detection increases the stability for increasing number of users. The baseline algorithm, however, suffers from frequent rate fluctuation, even at high k. It is interesting that the indoors-outdoors estimation shows a slight drawback for stability, but is insignificant for the other studied QoE factors. We can conclude that, compared to the baseline, *IOBBA* improves both streaming smoothness and stability by approximately 30% on average. This QoE improvement comes only at a marginal cost on the average rate.

Figure 4.8 to Figure 4.10 show the studied performance factors for an increasing target buffer level, while the number of users is kept constant at $k = 4$. A larger maximum buffer ensures that, in the segment-buffer map, the video rate levels are sufficiently distant from each other, thus reducing the probability of rate oscillation. Of course, the larger the maximum buffer level, the higher the average buffer occupancy per stream, and therefore, the lower the re-buffering frequency.

We should mention that the average rate, in buffer-based adaptation, is inversely proportional to the maximum buffer occupancy (see Figure 4.8); this is due to the fact that the segment maps of larger buffers have a lower response rate to throughput increases. The larger the buffer, the more data are necessary to be downloaded in order for $B_t > \theta_2$, thus achieving the maximum available rate. This explains the reduction of *IOBBA*'s average

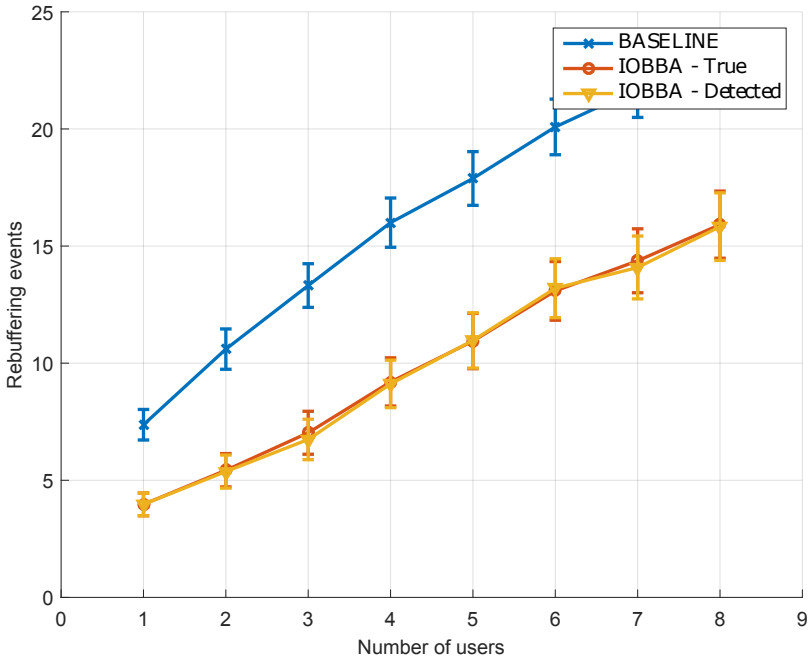


Figure 4.7: Re-buffering events vs. users for $B_{max} = 150s$

rate in Figure 4.8 (the linear segment map used by the baseline achieves faster increase in streamed rate, than *IOBBA*'s exponential segment map).

On the other hand, the smaller the maximum buffer occupancy, the smaller the distances between buffer values that indicate video rate switches and, thus, the larger the rate adaptation frequency; depicted in Figure 4.9. This is why *IOBBA* outperforms the baseline by 20 to 25% in stability and by 35 to 40% in terms of smoothness. All in all, this is a substantial QoE gain at a reasonable penalty for the average rate.

4.6 Conclusion

In this chapter, we extended a classic buffer-based rate adaptation algorithm by a Bayesian coverage estimator. The resulting *context-aware* rate adaptation algorithm, called *IOBBA*, aggressively reduces the requested video quality if the user is estimated to enter an area of poor signal coverage (e.g., indoors). Then the algorithm stays conservative, until an area of high coverage is detected (e.g., outdoors).

The coverage estimation requires no knowledge of user location and makes no use of coverage or map data. The required input is entirely collected during run time, from two smartphone sensors at the application layer. Estimation accuracy is in the order of 95%, which is sufficient for video

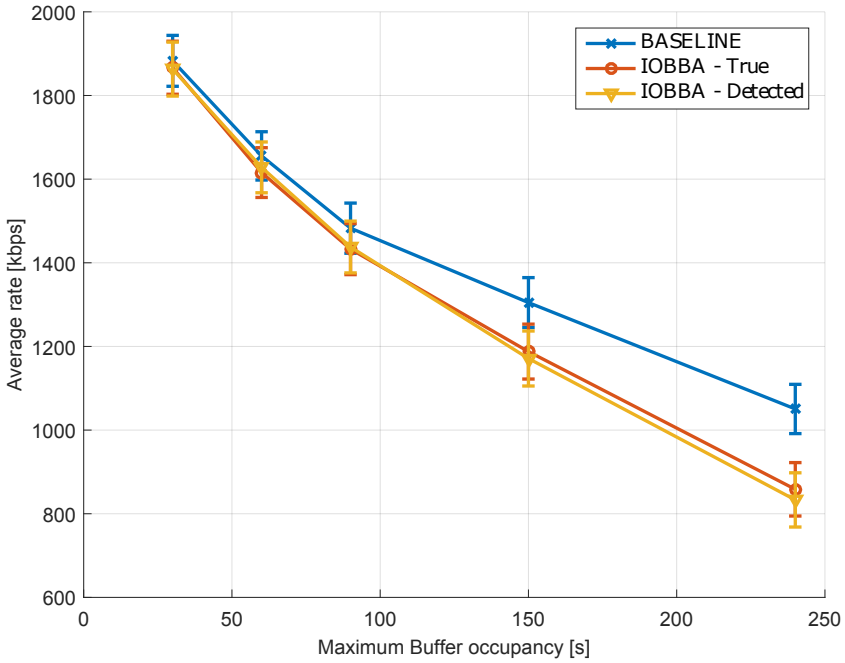


Figure 4.8: Average rate per B_{max} with 4 users

streaming QoE improvement.

Our experiments in a typical downtown office building show substantial improvements in video rate, smoothness and stability, compared to a conventional rate adaptation algorithm. This clearly demonstrates that making adaptive streaming aware of the user's current coverage situation can highly improve the QoE of mobile streaming.

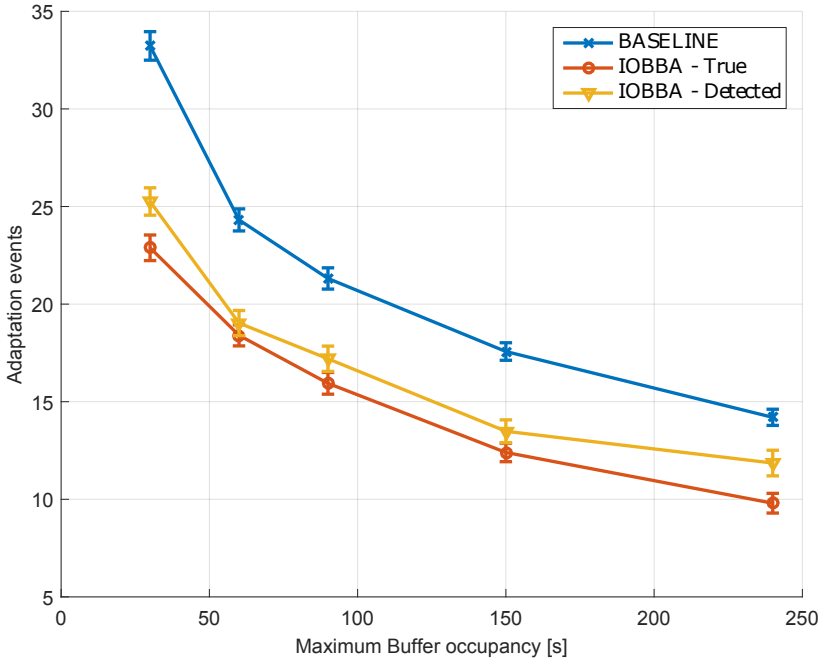


Figure 4.9: Adaptation events per B_{max} with 4 users

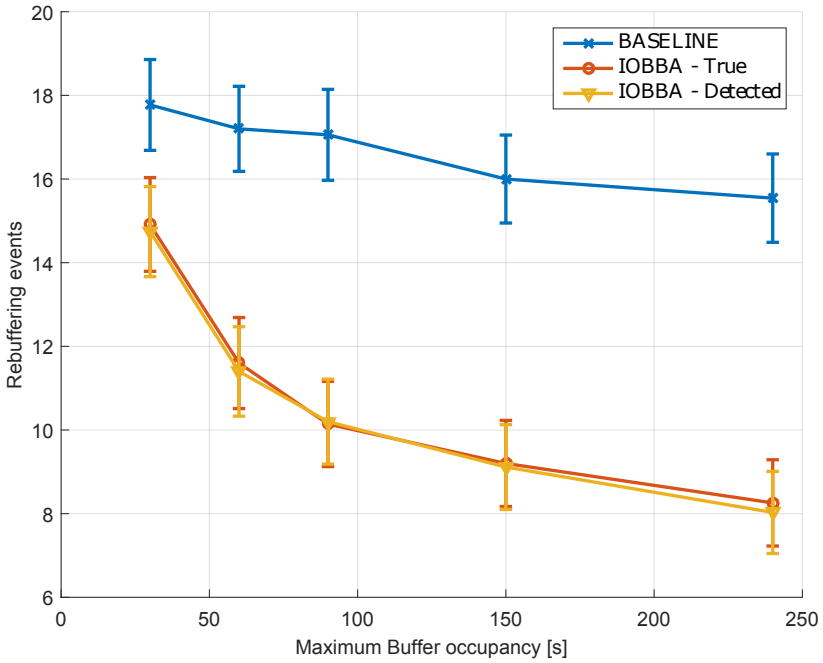


Figure 4.10: Re-buffering events per B_{max} with 4 users

Chapter 5

Adaptive streaming optimization via online-learning

5.1 Introduction

Constant developments in cellular networking technology, such as 4G's Long Term Evolution (LTE) or the anticipated 5G, are changing the landscape of mobile high-bandwidth multimedia applications, that are becoming fast an integral part of the mobile clients' life. In particular, the demand for *mobile* video streaming has been advancing at unprecedented growth rates over the last years. Nonetheless, cellular networks are typically characterized by throughput variation, caused by radio propagation effects. Such network conditions pose significant challenges on mobile streaming solutions, where optimal bitrate adaptation over fluctuating wireless channels remains a challenging operation. One key finding of our analysis in Chapter 3, argues that most existing rate adaptation schemes, may require parameter tuning according to the considered network profile or user scenario, and thus cannot generalize well beyond a certain scope of usage.

In an effort to overcome this limitation, some algorithms resort to learning techniques or control theoretic approaches [80] to attain optimal quality adaptation. However, their practical implementation on mobile devices may be hindered by energy-demanding architectures [137] or by the complexity of exploring the complete optimization space [138]. In this chapter we propose a novel adaptation algorithm based on OCO [56, 139], that is independent of any parameter selection concerning the streaming environment and does not require computationally heavy operations.

OCO has emerged as a very effective online learning framework, that is also suitable for mobile deployment, in terms of resource requirements. According to OCO, an agent learns to make sequential decisions in order

to minimize an adversarial loss function, unknown at decision time. OCO is "model-free", as no assumption for the statistical model of the channel is required, while at the same time it provides tractable feasibility and performance guarantees [140]. Having already been proposed for problems with rapidly fluctuating environments such as cloud resource reservation management [141] and dynamic network resource allocation [142], it constitutes an appealing candidate for optimization of HAS as well. However, the application of OCO in HAS is not a straightforward task. Given that HAS optimization has a discrete decision space (set of qualities for every segment) and instantaneous state-dependent constraints (finite-sized buffer queue), it does not fall directly in the class of OCO problems.

Nevertheless, after providing a set of relaxations and modelling assumptions, we propose *Learn2Adapt (L2A)*, a novel rate adaptation algorithm, that is shown to provide a *robust* adaptation strategy and which unlike most of the state-of-the-art techniques does not require parameter tuning, channel model assumptions, or application-specific adjustments. These properties make it very suitable for mobile users, who typically experience fast variations in channel characteristics. Simulations show that *Learn2Adapt* improves on average streaming bit-rate without impairing the overall QoE, a result that is independent of the channel and application scenarios.

5.1.1 Prior work

As we have covered in Chapter 2, primarily heuristic approaches have been proposed for client-side adaptation, that can be mainly classified into three categories according to the considered input. In Chapter 3 we made a per class evaluation of such methods. First, rate-based methods estimate the available channel rate to decide on the bit-rate of the streamed video. For instance, Li et al. [61] propose *PANDA*, an adaptation module that uses an exponential moving average filter to estimate the available bandwidth and schedules the download of every segment, in a way that reduces bit-rate oscillations, particularly in scenarios with multiple clients. A similar adaptation strategy, called *FESTIVE* [64], focuses primarily on fairness amongst all clients. Second, buffer-based methods use application-level signals, such as the instantaneous buffer level to perform the adaptation. A notable such method comes from Huang et al. [68], who propose *BBA*, that follows a mapping between instantaneous buffer values to bit-rate levels. Third, hybrid methods may use a combination of inputs. In that direction, Kim et al. [143] propose *XMAS*, a hybrid adaptation method that deploys a client-based traffic shaping scheme to throttle the server's packet transmission, based on both bandwidth estimates and playback buffer levels, while Xie et al. [124] propose *piStream*; a physical-layer informed adaptation strategy. Lately, new smartphone devices have enabled the fusion of multiple sensor readings to infer context in the mobile client's environment. To this end,

in Chapter 4 we solicited incorporating a user's inferred location into the decision process.

Recently, there has been a shift in the scientific literature, in regard to the methods used in the rate adaptation design; primarily towards optimization and control theoretic approaches. In that direction, Spiteri et al. [69] formulate rate adaptation as a utility maximization problem and devise *BOLA*, an online control algorithm, that makes use of the instantaneous buffer occupancy. *MPC* by Yin et al. [138] is another notable control-theoretic scheme, that combines buffer occupancy and throughput predictions for optimal rate adaptation.

In regard to rate adaptation methods based on optimization and in particular on dynamic programming, Zhou et al. [144], propose *mDASH* and formulate the rate adaptation logic as a MDP where the buffer size, bandwidth conditions and bitrate stability are taken as Markov state variables. Similarly, Bokani et al. [81] model the rate adaptation logic as an MDP problem as well and incorporate mobility by including vehicular environments. Some of the main drawbacks of MDP-based solutions are computational load and the need to know the statistics of the network and video content in advance.

Model-free Reinforcement Learning (RL) approaches, such as Q-Learning (QL), have also been investigated for the design of rate adaptation methods. Claeys et al. [145] propose a QL-based HAS client, allowing dynamical adjustment of the streaming behavior to the perceived network state. While QL approaches provably converge to the optimal policy, provided that their parameters are chosen correctly, the convergence speed becomes an issue when trying to cope with previously unseen channel or video content patterns.

Lately, Deep-Learning (DL) approaches have also been proposed for rate adaptation, presenting promising merits in both accuracy and convergence of bitrate decisions. *Pensieve* [146] is a rate adaptation DL framework that does not rely on pre-programmed models or assumptions about the environment, but instead gradually learns the best policy for bitrate decisions through observation and experience. Another DL approach is called *D-DASH* [137], that combines DL and RL mechanisms and achieves a good trade-off between policy optimality and convergence speed during the decision process. Nonetheless, the deployment of DL in mobile devices is typically associated with high computational and energy demands, especially during training phases and thus external (hardware) resources may be required to assist in the rate adaptation process.

Huang et al. [147] have recently explored combinatorial optimization for rate adaptation and have proposed *Hindsight*, a near-optimal, linear-time and linear-space greedy algorithm.

During our literature review, we have identified a requirement for a rate adaptation approach, that is not only scalable to the number of clients, but also light in computation and that does not rely on any modelling

assumptions. We attempt to fulfill these prerequisites with our novel method proposed in Section 5.3. A more detailed survey on adaptive streaming solutions can be found in [60] and [58].

5.1.2 Contributions

The work presented in this chapter, provides multiple contributions towards formulating the HAS optimization problem under the OCO framework:

- we model the adaptive streaming client by a learning agent, whose objective is to maximize the average video rate of a streaming session, subject to scheduling constraints of the buffer queue. In general, the choice of the objective function is made under the assumption that higher average video rate typically corresponds to higher video quality, when comparing the same video and the same resolution. The constraints are chosen in consideration of the buffer queue, since re-buffering events can significantly influence QoE [48]
- We fulfill the OCO requirement that both the set of decisions and constraint functions must be convex by:

    1. allowing the agent to make decisions on the video quality of each segment, according to a probability distribution and by

    2. deriving a set of convex constraints associated with the upper and lower bound of the finite buffer queue. We achieve the latter by making a relaxation to an unbounded buffer that adheres to time-averaging constraints.

- We model the channel rate evolution by an adversary, that decides the cost of each decision only after it has been taken.
- We eventually solve the HAS optimization problem by proposing *Learn2Adapt*, a novel, online-learning, adaptation algorithm based on the OCO theory.
- In our trace-based simulations, our proposed method proves to be *robust*, providing consistently better QoE, when evaluated against reference state-of-the-art adaptation algorithms in a wide spectrum of possible network and streaming conditions.

5.2 System Model

This section introduces the model for the media content and client operations used in the rest of this work. Moreover, the notation is summarized in Table 5.1.

5.2.1 Media model

Let us assume that a video sequence of duration D seconds is stored on a server organized in the form of $T = \lceil D/V \rceil$ segments, each of constant playback duration V. Each segment is encoded at N quality representations at increasing *target bitrate* $r \in \{r_1, \ldots, r_N\}$. For a given quality representation $n \in \{1, \ldots, N\}$, the *actual size* of the t-th segment ($t \in \{1, \ldots, T\}$) – denoted $S_{t,n}$ and measured in bits – is a function of the segment content. In the following, we will assume that the server is connected to the client across a channel of rate C_t and thus the t-th segment is downloaded across the channel in $\frac{S_{t,n}}{C_t}$ seconds.

5.2.2 Client model

The client issues a request to the server, for the t-th segment and then waits for that segment to be fully downloaded before requesting the $(t+1)$-th segment. We refer to the, typically variable, interval between two consecutive requests as a *decision epoch*. Since the content is downloaded in T segments, the total number of decision epochs is T and referred to as the *horizon*. At the beginning of the t-th epoch, the client selects the quality representation $x_t \in \mathcal{X} = \{1, \ldots N\}$ for segment t, corresponding to the bitrate indication $r_{x_t} \in \{r_1, \ldots, r_N\}$ of the deployed rate adaptation algorithm.

Let B_t represent the buffer level at the beginning of the t-th epoch, measured in seconds of buffered video at the client. The downloaded segments are stored in a buffer whose size may not exceed an upper bound B_{max}, that exists normally due to memory constraints of the mobile device. Upon completely downloading the t-th segment, B_t increases by V seconds.

However, due to the concurrent playback of the buffered segments, B_t will also decrease by the amount of time required to download the t-th segment, which is equal to $\frac{S_{t,x_t}}{C_t}$ seconds (as long as $B_t > 0$). So, the buffer level evolves between two consecutive epochs according to:

$$B_{t+1} = \left[B_t - \frac{S_{t,x_t}}{C_t}\right]^+ + V - \Delta_t, \tag{5.1}$$

where $[x]^+ \triangleq \max(0, x)$. A delay $\Delta_t = \left[B_t - \frac{S_{t,x_t}}{C_t} + V - B_{max}\right]^+$ is introduced to account for the upper bound B_{max} of the buffer size. In other words, if $B_t - \frac{S_{t,x_t}}{C_t} + V < B_{max}$, the $(t+1)$-th segment is requested immediately and $\Delta_t = 0$. Otherwise, the request for the $(t+1)$-th segment is delayed by Δ_t seconds, to allow the buffer to drop to B_{max}. This delay protects against *buffer overflow* incidents, which occur when the buffer surpasses B_{max} and creates the characteristic bursty traffic of HAS [148]. A *buffer underflow* occurs when the instantaneous buffer level drops below zero, causing a *stall* in the video playback, an event that significantly degrades the QoE [48].

Table 5.1: Notations

Notation	Definition	Units
D	Video content total duration	seconds
V	Segment duration	seconds
T	Streaming horizon	segments
N	Quality representations	scalar
x_t	Selected quality representation for segment t	scalar
r_{x_t}	Bitrate corresponding to quality x_t	kbps
$S_{t,n}$	File size of segment t in n-th quality	kbits
ω_t	Decision distribution	probability vector
ω^*	Benchmark distribution	probability vector
Q	Virtual queue	scalar
V_L	Cautiousness parameter	scalar
α	Step-size	scalar
β	Target switching rate	switches per epoch
γ	Switch counter	scalar
C_t	Channel rate at epoch t	kbps
B_t	Buffer level at epoch t	seconds
B_{max}	Maximum buffer level	seconds
Δ_t	Buffer delay	seconds

In the next section we provide a framework which allows us to design a learning algorithm, that provably optimizes video quality subject to keeping the buffer asymptotically away from the two limits.

5.3 Adaptive streaming problem formulation

This section provides an algorithmic solution based on the theory of OCO [139]. In order to cast the video streaming optimization problem as an OCO with budget constraints problem, we first propose a relaxation on the finite buffer queue and then we modify the formulation to convexify the decision space. In the following, we present our online-learning algorithm *Learn2Adapt* (*L2A*), based on gradient descent and we provide theoretical guarantees for its performance.

5.3.1 OCO formulation

We formulate the rate adaptation problem as a *constrained OCO* problem, where the goal is to minimize the cumulative losses $\sum_{t=1}^{T} f_t(x_t)$ (referring to the average bitrate of the downloaded segments) while keeping the cumulative constraint functions $\sum_{t=1}^{T} g_t^i(x_t), \ \forall i = 1, 2$, negative (referring to buffer underflow and overflow); see also the relevant literature [149, 150]. In the OCO framework, functions $f_t, g_t^i \ \forall i = 1, 2$ are chosen by an *adversary* and are unknown at decision time. We will relate these functions to the

random evolution of the channel rate C_t, which in nature is not adversarial. Nevertheless, the adversarial setting is more general and includes any – potentially time-varying – distribution of C_t, which in turn bestows on our algorithm superior *robustness*. Next, we explain how these functions are used in our system.

Recall the set of quality representations $\mathcal{X}$, and let $x_t \in \mathcal{X}$ be the decision for the quality representation of the segment to be downloaded in epoch t. Consider the following functions:

$$\tilde{f}_t(x_t) \triangleq -r_{x_t} \tag{5.2}$$

$$\tilde{g}_t^1(x_t) \triangleq \frac{S_{t,x_t}}{C_t} - V, \tag{5.3}$$

$$\tilde{g}_t^2(x_t) \triangleq V - \frac{S_{t,x_t}}{C_t} - \frac{B_{max}}{T}, \tag{5.4}$$

where (5.2) captures the utility (higher bitrate yields smaller losses). (5.3)-(5.4) express the buffer displacement, which will be used to model the buffer underflow and overflow constraints, respectively. A high quality representation x_t combined with a low channel rate C_t will prolong download time $\frac{S_{t,x_t}}{C_t}$, which will result in high buffer consumption. Since C_t is unknown at decision time of x_t, it is impossible to know the values of $\tilde{g}_t^i(x_t),\ \forall i = 1, 2$. Our approach therefore, is to learn the best x_t based on our estimation of $\tilde{g}_t^i(x_t),\ \forall i = 1, 2$.

To cast the above problem as OCO with budget constraints, we propose the following steps:

- First, we provide a relaxation to the hard constraints of the buffer model.
- Second, we convexify the decision set by randomization. We associate a probability to each decision, and we learn the optimal probability distribution for deciding the quality representation to download at each epoch.

5.3.2 Buffer constraints

Here we explain how we use the cumulative constraint functions $\sum_{t=1}^{T} \tilde{g}_t^i(x_t)$, $\forall i = 1, 2$ to model buffer underflow and overflow, respectively. The buffer evolves according to (5.1) and ensuring $0 \leq B_t \leq B_{max},\ \forall t$, involves in principle a very complicated control problem, which in the presence of unknown adversarial C_t is exacerbated.

To avoid computationally heavy approaches and to arrive at a simple (yet robust) solution, we thus seek an alternative approach. In that direction, we treat the buffer as an infinite queue, with the simpler (compared to (5.1)) update rule: $B_{t+1} = B_t + V - S_{t,x_t}/C_t$, where now no additional delay Δ_t is ever imposed on the system. By this, we allow instantaneous

violation of the budget, but we utilize a penalty which aims to maintain the buffer on the $[0, B_{max}]$ range on average. In particular, using (5.3)-(5.4), we capture in $\tilde{g}_t^i(x_t),\ \forall i = 1, 2$ the instantaneous buffer displacement on both directions (measured in seconds) and by requiring the cumulative constraint $\sum_{t=1}^{T} \tilde{g}_t^i(x_t) \leq 0,\ \forall i = 1, 2$, we ensure that on average B_t remains in the non-negative regime below B_{max}. A benefit is that these constraints are in the realm of OCO theory, and therefore allow us to design a simple learning algorithm that provably satisfies them. Overall, our approach here is to apply a loosely coupled control to the buffer constraints, by tolerating instantaneous violations and ensuring that in the long-term only a few are experienced.

5.3.3 Convexification

To obtain a convex decision set, we use a convexification method based on randomization of the decision process [151]. Consider the probability simplex:

$$\Omega = \{\boldsymbol{\omega} \in \mathbb{R}^N : \boldsymbol{\omega} \geq 0 \wedge \|\boldsymbol{\omega}\|_1 = 1\},$$

where $\omega_n = \mathbb{P}(x = n)$ denotes the probability that we decide $x = n \in \{1, \ldots, N\}$ and Ω is a convex set. Thus, instead of learning directly the decision x_t, we learn the optimal probability $\boldsymbol{\omega}_t = (\omega_{t,n})_{n=1,\ldots,N}$ of picking x_t from the integer set $\mathcal{X}$. Given a decision $\boldsymbol{\omega}_t$, the actual quality representation will be chosen according to the expectation of the corresponding utility, i.e. $x_t \in \arg\min_{x \in \mathcal{X}} |r_x - \sum_{n=1}^{N} \omega_{t,n} r_n|$. The functions of interest become now random processes and we must appropriately modify them by taking expectations with respect only to $\boldsymbol{\omega}_t$ and not to the randomness of C_t:

$$f_t(\boldsymbol{\omega}_t) \triangleq -\mathbb{E}[r_{x_t}] = -\sum_{n=1}^{N} \omega_{t,n} r_n \tag{5.5}$$

$$g_t^1(\boldsymbol{\omega}_t) \triangleq \mathbb{E}\left[\frac{S_{t,x_t}}{C_t} - V\right] = \frac{\sum_{n=1}^{N} \omega_{t,n} S_{t,n}}{C_t} - V \tag{5.6}$$

$$g_t^2(\boldsymbol{\omega}_t) \triangleq \mathbb{E}\left[V - \frac{S_{t,x_t}}{C_t} - \frac{B_{max}}{T}\right] = V - \frac{\sum_{n=1}^{N} \omega_{t,n} S_{t,n}}{C_t} - \frac{B_{max}}{T}. \tag{5.7}$$

Given the loss function and constraints above, we formulate the constrained OCO problem, that we solve in Section 5.4:

$$\min_{\boldsymbol{\omega} \in \Omega} \sum_{t=1}^{T} f_t(\boldsymbol{\omega}) \quad \text{s.t.} \quad \sum_{t=1}^{T} g_t^i(\boldsymbol{\omega}) \leq 0 \quad \forall i = 1, 2.$$

The following are true for functions (5.5)-(5.7) and our surrogate convex problem:

- The diameter of Ω, defined as the largest Euclidean distance between any two vectors, is $\sqrt{N}$.
- Functions f_t and $g_t^i,\ \forall i = 1, 2$ are smooth, bounded and have bounded gradients. Specifically, $\forall t, \boldsymbol{\omega}, i = 1, 2$:

$$
\begin{aligned}
&|f_t(\boldsymbol{\omega})| \leq r_N, \\
&|g_t^1(\boldsymbol{\omega})| \leq \max\left\{\left|\frac{S_{\min}}{C_{\max}} - V\right|, \left|\frac{S_{\max}}{C_{\min}} - V\right|\right\}, \\
&|g_t^2(\boldsymbol{\omega})| \leq \max\left\{\left|V - \frac{S_{\min}}{C_{\max}} - \frac{B_{max}}{T}\right|, \left|V - \frac{S_{\max}}{C_{\min}} - \frac{B_{max}}{T}\right|\right\}, \\
&\|\nabla f_t(\boldsymbol{\omega})\| \leq \sqrt{\sum_{n=1}^{N} r_n^2}, \quad \|\nabla g_t^i(\boldsymbol{\omega})\| \leq \sqrt{\sum_{n=1}^{N}\left(\frac{S_{t,n}}{C_{\min}}\right)^2},
\end{aligned}
$$

where $C_t \in [C_{\min}, C_{\max}]$, $S_{t,j} \in [S_{\min}, S_{\max}]$ and $\nabla f_t(\boldsymbol{\omega}), \nabla g_t^i(\boldsymbol{\omega}),\ \forall i = 1, 2$ denote the gradients.

5.3.4 Regret metric

At every decision epoch $t = 1, 2, \ldots T$ the following events occur in succession:

(a) the agent computes $\boldsymbol{\omega}_t \in \Omega$ according to an algorithm,

(b) the agent chooses $x_t \in \arg\min_{x \in \mathcal{X}} |r_x - \sum_{n=1}^{N} \omega_{t,n} r_n|$,

(c) an adversary decides C_t, and the loss function $\tilde{f}_t(\boldsymbol{\omega}_t)$ and the constraint functions $\tilde{g}_t^i(\boldsymbol{\omega}_t),\ \forall i = 1, 2$ are determined using (5.2)-(5.4), and then used to measure the actual loss and buffer displacement,

(d) the following forms of feedback are provided to the agent: (i) the value of C_t, (ii) the functions $f_t, g_t^i,\ \forall i = 1, 2$, (iii) the gradients $\nabla f_t(\boldsymbol{\omega}_t), \nabla g_t^i(\boldsymbol{\omega}_t),\ \forall i = 1, 2$.

The feedback above is used by the agent to eventually determine the gradient vectors $\nabla f_{t+1}(\boldsymbol{\omega}_{t+1}), \nabla g_{t+1}^i(\boldsymbol{\omega}_{t+1}),\ \forall i = 1, 2$. We now define the performance metric in our problem which consists of two parts: the *regret* of an algorithm and the i-th *constraint residual*, defined as:

$$
R_T = \sum_{t=1}^{T} f_t(\boldsymbol{\omega}_t) - \sum_{t=1}^{T} f_t(\boldsymbol{\omega}^*) \quad \text{and} \quad V_T^i = \sum_{t=1}^{T} g_t^i(\boldsymbol{\omega}_t),
$$

respectively. Here $\boldsymbol{\omega}^* \in \Omega$ is a benchmark distribution, that minimizes the losses in hindsight, with knowledge of the functions $f_t, g_t^i,\ \forall i = 1, 2$. This benchmark satisfies the cumulative constraints every K:

$$
\begin{aligned}
\boldsymbol{\omega}^* \in \arg\min_{\boldsymbol{\omega} \in \Omega} & \sum_{t=1}^{T} f_t(\boldsymbol{\omega}) \\
\text{s.t.} \quad & \sum_{t=k}^{K+k-1} g_t^i(\boldsymbol{\omega}) \leq 0, \\
& \forall k = 1, \ldots, T - K + 1, \quad \text{and} \quad \forall i = 1, 2.
\end{aligned}
$$

This benchmark is first explained in [149], where the authors prove that for any $K = o(T)$, a smart agent can learn to have no regret, while satisfying the adversarial constraints. In our case, picking $K = T^{1-\epsilon}$, for small $\epsilon > 0$, gives the best approximation of our algorithms' performance, allowing maximum freedom to the competing benchmark. If an algorithm achieves both $o(T)$ regret and $o(T)$ constraint residual, then it follows that as $T \to \infty$ we have (i) $R_T/T \to 0$, hence our algorithm has the same losses with (or "learns") the benchmark action, and (ii) $V_T^i/T \to 0,\ \forall i = 1, 2$, hence our algorithm ensures the average constraint. Since the benchmark action is the best *a posteriori* action, taken with knowledge of all the revealed values of C_t, learning it is both remarkable and very useful.

5.4 OCO solution

In this section, we propose a "no regret" algorithm to solve the constrained OCO problem defined in the previous section. We first provide the intuition behind the algorithm design and the introduction of a switching budget, that allows the control of the switching frequency for our algorithm. We then detail the proposed algorithm, and finally provide some performance bounds.

5.4.1 Learn to Adapt (L2A) algorithm

As a general note, a main challenge in such problems is that the constraints $g_t^i(\boldsymbol{\omega}_t),\ \forall i = 1, 2$ are not known when the decision of $\boldsymbol{\omega}_t$ is taken. The OCO approach to this issue is to predict such functions using a first order Taylor expansion of $g_t^i(\boldsymbol{\omega}_t),\ \forall i = 1, 2$ around $\boldsymbol{\omega}_{t-1}$ evaluated at $\boldsymbol{\omega}_t$ [139]:

$$\hat{g}_t^i(\boldsymbol{\omega}_t) \triangleq g_{t-1}^i(\boldsymbol{\omega}_{t-1}) + \langle \nabla g_{t-1}^i(\boldsymbol{\omega}_{t-1}), \boldsymbol{\omega}_t - \boldsymbol{\omega}_{t-1} \rangle,\ \forall i = 1, 2. \tag{5.8}$$

We recall that in (5.8), only $\boldsymbol{\omega}_t$ is unknown at t, whereas $\boldsymbol{\omega}_{t-1}$, $\nabla g_{t-1}^i(\boldsymbol{\omega}_{t-1})$ and $g_{t-1}^i(\boldsymbol{\omega}_{t-1}),\ \forall i = 1, 2$ are known via the obtained feedback.

Contrary to the standard (unconstrained) online gradient [139], our algorithm must combine the objective and the constraint functions. To this end, consider the regularized Lagrangian:

$$L_t(\boldsymbol{\omega}, \boldsymbol{Q}(t)) = \sum_{i=1}^{2} Q_i(t) \hat{g}_t^i(\boldsymbol{\omega}) + V_L \hat{f}_t(\boldsymbol{\omega}) + \alpha ||\boldsymbol{\omega}_t - \boldsymbol{\omega}_{t-1}||^2, \tag{5.9}$$

where $Q_i(t)$ is the Lagrange multiplier, $\hat{g}_t^i(\boldsymbol{\omega})$ is the prediction of the constraint function $g_t^i(\boldsymbol{\omega})$ from (5.8), V_L is a *cautiousness* parameter that controls the trade-off between regret and constraint residual, $\hat{f}_t(\boldsymbol{\omega})$ applies (5.8) to f_t, α is the step-size and $||\boldsymbol{\omega}_t - \boldsymbol{\omega}_{t-1}||^2$ is a regularization term that smooths the decisions. Parameters V_L and α are tuned for convergence and their choices are given below. We mention here, that the Lagrange multiplier

$Q_i(t),\ \forall i = 1, 2$ is updated in a *dual ascent* approach, by accumulating the constraint deviations:

$$Q_i(t+1) = [Q_i(t) + \hat{g}_t^i(\boldsymbol{\omega}_t)]^+,\ \forall i = 1, 2. \tag{5.10}$$

We further compound the online optimization problem by introducing a switching budget. Let $\beta \in (0, 1]$ be the maximum allowed reconfiguration frequency measured in quality switches per epoch. The goal is to limit the number of changes within the horizon to at most βT[1]. This is a valuable property that allows stability control for the following algorithm, that takes a step in the direction of the sub-gradient of the regularized Lagrangian. The resulting algorithm is reported in pseudo-code in Algorithm 1.

Algorithm 1 Learn2Adapt (L2A)

Initialize: $\boldsymbol{Q}(1) = 0$, $\boldsymbol{\omega}_0 \in S$, $t' = 1$
Parameters: cautiousness parameter V_L, step size α, maximum allowed switch rate β, switch counter $\gamma = 0$
1: **for** all $t \in \{1, 2, \ldots, T\}$ **do**
2: **if** $\frac{\gamma}{t} \leq \beta$ **then**
3: $\boldsymbol{\omega}_t = \text{proj}_{\Omega}\left[\boldsymbol{\omega}_{t-1} - \frac{\sum_{j=t'}^{t}\{V_L \nabla f_{j-1}(\boldsymbol{\omega}_{j-1}) + \sum_{i=1}^{2} Q_i(j) \nabla g_{j-1}^i(\boldsymbol{\omega}_{j-1})\}}{2\alpha}\right]$
4: $t' = t + 1$
5: $\gamma + +$
6: **else**
7: $\boldsymbol{\omega}_t = \boldsymbol{\omega}_{t-1}$
8: **end if**
9: $Q_i(t+1) = [Q_i(t) + \hat{g}_t^i(\boldsymbol{\omega}_t)]^+,\ \forall i = 1, 2$
10: **end for**

Here $\text{proj}_{\Omega}[\cdot]$ denotes the Euclidean projection on set Ω.

5.4.2 Performance guarantees

The main contribution of this work is the formulation of the rate adaptation problem in the constrained OCO framework and the proposal of *Learn2Adapt* (*L2A*). In the following we invoke the theorem from [149], to provide theoretical performance guarantees for the *Learn2Adapt* algorithm. We note here that although the following theoretical guarantees are derived for $\beta = 1$, in the numerical evaluation of Section 5.5 we provide evidence that *L2A* performs well even for $\beta < 1$.

Theorem 1 (From [149]). *For $\beta = 1$, choose small $\epsilon > 0$, fix $K = o(T^{1-\epsilon})$, $V_L = T^{1-\epsilon/2}$, and $\alpha = V_L\sqrt{T}$. Then, the* Learn2Adapt (L2A) *algorithm*

[1]While β may allow a switch at a given epoch t, $\boldsymbol{\omega}_t = \boldsymbol{\omega}_{t-1}$ is still a valid decision.

guarantees:

$$R_T = O(T^{1-\epsilon/2}), \qquad V_T^i = O(T^{1-\epsilon/4}),\ \forall i = 1, 2.$$

Effectively, this means that over time our algorithm learns the best a-posteriori distribution $\boldsymbol{\omega}^*$, which neatly satisfies the average constraints and minimizes the cumulative quality losses. We experimentally verify below that the corresponding choices x_t made by sampling this distribution have extremely well performing properties for video streaming adaptation.

5.5 Experimental Evaluation

In this section we evaluate the performance of our proposed rate adaptation algorithm against two reference rate adaptation schemes, by experimenting with real mobile network traces and video sequences, for two separate streaming applications (VOD and live streaming) and under five video streaming performance metrics.

5.5.1 Experimental setup

Network scenarios
In our evaluation, we use *real* cellular network traces and in particular a dataset that includes 4G channel measurements for various mobility scenarios [152]. For our experiments, we have selected the *static*, *pedestrian* and *car* scenarios (operator A therein), as realistic cases for no, low and high mobility, respectively. The {*static*, *pedestrian*, *car*} scenario consist of $\{12, 26, 41\}$ traces with an average measurement duration of $\{17, 18, 23\}$ min, respectively. Cellular networks present significant challenges to the rate adaptation process, as they are typically characterized by rapid throughput fluctuation and short service outages; that may be caused by radio propagation effects, low-coverage areas or handover events. While these characteristics are realistically depicted in the selected traces of [152], taking a step further in our evaluation, we have designed a *synthetic* scenario consisting of 20 traces, that is characterized by abrupt and steep channel rate transitions. This so-called *markovian* scenario emulates two channel levels (states) $\{0.75, 23.0\}$ Mbps with a 0.05 state transition probability and is complementary to the real traces; to present the rate adaptation algorithms with an additional, even more demanding network scenario.

Video parameters
In [153] video sequences are encoded at multiple bitrates in conditions typical of OTT video delivery. We used 3 sequences: *BBB*, *TOS* and *Sintel*, encoded in the H.264/AVC standard, at target bitrates $\{0.37, 0.75, 1.5, 3.0, 5.8, 12.0, 17.0, 20.0\}$ Mbps, corresponding to resolutions in $\{384\times216, 640\times360, 1024\times576, 1280\times$

$720, 1920\times1080, 3840\times2160\}$, and organized in DASH segments with duration $V = 2$s.

Streaming scenarios In our experiments, we consider a VOD streaming scenario and a live streaming scenario. For the VOD scenario, we considered a maximum buffer value of $B_{max} = 120$s (60 segments). For the live streaming scenario, we reduced the maximum buffer value to $B_{max} = 20$s (10 segments), according to the tighter latency requirements. All the figures below concern the case of VOD, while the results for the live streaming scenario are presented in Table 5.3.

Algorithms

We compare our method *L2A*, for $\beta = 0.3$ and $\beta = 1$, against *RB*, a throughput-based method and *BB*, a buffer-based method, following the design principles and parameters selection found in [61] and [69], respectively.

RB [61] is a very common throughput-based rate adaptation scheme based on a four-step adaptation model, where initially the available network bandwidth is estimated using a proactive probing mechanism, that is designed to minimize bitrate oscillations. Then the throughput estimates are smoothed using noise-filters to avoid estimation errors due to throughput variation. Next in the scheduling step, a sophisticated scheduler is considered that drives the buffer level towards the maximum buffer length B_{max}. At the same time the inter-request time is matched to the necessary time needed to complete the download based on the smoothed estimated value of the available bandwidth.

BB [69] is a buffer-based rate adaptation algorithm that uses Lyapunov optimization in order to indicate the bitrate of each segment. Practically, the algorithm is designed to maximize a joint utility function that rewards an increase in the average bitrate and penalizes potential stalls. More specifically the implemented variation, called *BOLA-O*, mitigates bitrate oscillations by introducing a form of bitrate capping when switching to higher bitrates.

These rate adaptation methods are widely used in research, each amongst the best performing methods of their class [**?**]. Regarding our method *L2A*, the presented results consider a cautiousness parameter of $V_L = T^{0.9}$ and step size of $\alpha = V_L\sqrt{T}$. We note here that according to DASH, in case of a stall, τ segments must be downloaded in order for the play-out to resume. For all algorithms we considered $\tau = 2$.

Video streaming performance metrics

We evaluate the performance of our proposed method based on the video streaming performance metrics presented in Table 5.2. Although these metrics are very similar to those used in Chapter 3, here we use a slightly different normalization scheme. *Average bitrate* models the average bitrate

Table 5.2: Video streaming performance metrics

Metric name	Element evaluated	Metric
Average bitrate	Average video bitrate	$\frac{\bar{r}}{\max_{m\in\mathcal{M}}\bar{r}^m}$
Stability	Bitrate switching frequency	$1-\frac{\sum_{t=2}^{T}\left(\mathbb{1}_{\{r_{x_t}\neq r_{x_{t-1}}\}}\right)}{T-1}$
Smoothness	Adaptation amplitude	$1-\frac{\sum_{t=2}^{T}\lvert r_{x_t}-r_{x_{t-1}}\rvert}{(r_N-r_1)(T-1)}$
Consistency	Stall duration	$1-\frac{\sum_{t=1}^{T}\mathbb{1}_{\left\{B_{t-1}<\frac{S_{t,x_t}}{C_t}\right\}}\left(-B_{t-1}+\sum_{k=0}^{\tau-1}\frac{S_{t+k,x_{t+k}}}{C_{t+k}}\right)}{D}$
Continuity	Frequency of stalls	$1-\frac{\sum_{t=1}^{T}\mathbb{1}_{\left\{B_{t-1}<\frac{S_{t,x_t}}{C_t}\right\}}}{\lceil\frac{T}{\tau}\rceil}$

Table 5.3: Live streaming ($B_{max}=20s$) results (BB / RB / L2A ($\beta=0.3$) / L2A ($\beta=1$))

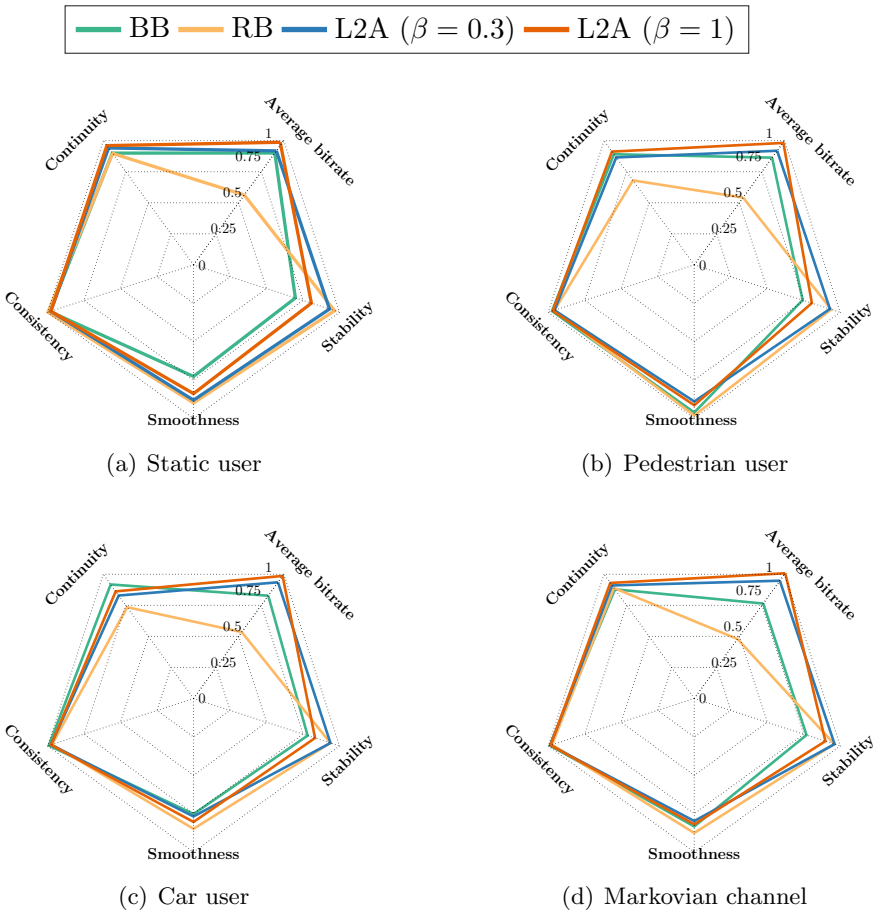
	Static	Pedestrian	Car	Markovian
Average bitrate	0.93 / 0.59 / 0.88 / **0.98**	0.94 / 0.58 / 0.93 / **0.96**	0.93 / 0.59 / 0.96 / **0.98**	0.91 / 0.69 / 0.97 / **1.00**
Stability	0.42 / **0.95** / 0.82 / 0.72	0.50 / **0.92** / 0.86 / 0.75	0.56 / **0.91** / 0.86 / 0.78	0.86 / **0.93** / 0.87 / 0.82
Smoothness	0.86 / 0.92 / 0.94 / **0.94**	0.86 / 0.92 / 0.94 / **0.95**	0.87 / 0.95 / 0.94 / **0.97**	0.96 / 0.95 / 0.92 / **0.98**
Consistency	0.87 / 0.81 / 0.92 / **0.92**	0.85 / 0.62 / 0.83 / **0.85**	0.88 / 0.62 / 0.80 / **0.85**	0.78 / 0.48 / 0.83 / **0.84**
Continuity	0.93 / 0.95 / 0.97 / **0.97**	0.93 / 0.93 / 0.97 / **0.97**	0.93 / 0.94 / 0.95 / **0.95**	0.92 / 0.88 / 0.94 / **0.94**

$\bar{r}=\frac{\sum_{t=1}^{T}r_{x_t}}{T}$ of the received video segments in a session, normalized over the maximum average bitrate $\max_{m\in\mathcal{M}}\bar{r}^m$ obtained for that session by any adaptation method $m\in\mathcal{M}$, where $\mathcal{M}$ is the set of all evaluated methods. Streaming *stability* models the frequency of bitrate switching, while streaming *smoothness* is associated with the amplitude of the bitrate switches, i.e the absolute bitrate difference between sequential segments. Both *stability* and *smoothness* are normalized over the maximum attainable value for each respective metric, while $\mathbb{1}_{\{\mathcal{Y}\}}$ is an indicator vector; with ones at the positions that condition $\mathcal{Y}$ is true, and zeros otherwise. Additionally, we propose two metrics associated with a) the frequency of stalls and b) their severity (duration). With streaming *consistency* we measure the percentage of the user's allocated time-budget (typically equal to the video length D) that was spent actually consuming video content (as opposed to stalling), while streaming *continuity* expresses the percentage of segments that were downloaded while play-out remained uninterrupted, assuming $B_0=0$.

5.5.2 Results

In regard to the evaluation results, each point on Figure 5.1 corresponds to a score for each of the five performance metrics and is the average result over all traces for the considered network scenario. In particular, Figure 5.1(a) shows the performance of a static user (no mobility), Figure 5.1(b) shows the performance of a pedestrian user (low mobility), Figure 5.1(c) shows a user while being mobile in a car (high mobility) and Figure 5.1(d) corresponds to the artificial *markovian* scenario.

In Figure 5.1 *L2A*, our proposed method, registers significant improvement in average bitrate, almost up to 45% against *RB* and up to 20% against *BB*,



(a) Static user

(b) Pedestrian user

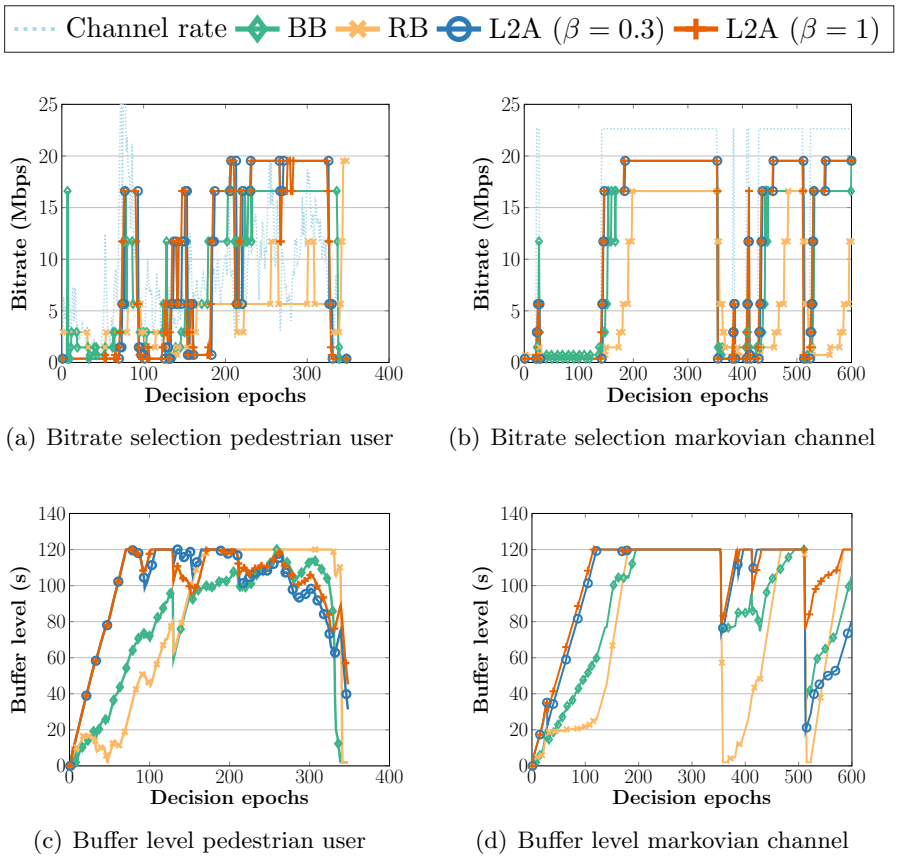
(c) Car user

(d) Markovian channel

Figure 5.1: Performance evaluation results - *L2A* improves average bitrate

for all studied real network scenarios and for both the cases of restricted ($\beta = 0.3$) and unrestricted ($\beta = 1$) switching. At the same time *L2A* offers consistent (i.e without interruptions) streaming with equivalent continuity to all the other methods, i.e. all methods experience a few brief stalls during periods of very poor channel quality. In regard to smoothness, all methods obtain equivalent scores. Nonetheless, in regard to the stability metric, *L2A*'s restricted switching variant ($\beta = 0.3$) achieves about 15% improvement in stability when compared to the case of unrestricted ($\beta = 1$) switching; a result that is anticipated from our algorithmic design. Additionally, *L2A*'s restricted switching variant ($\beta = 0.3$) improves on stability by 25% against *BB*. Comparing *L2A* and *RB* in terms of stability, we observe that although both perform well in this metric, *RB* is overall more conservative, given the low average bitrate it obtained in all scenarios.

In the markovian network scenario depicted in Figure 5.1(d), *L2A* performs 50% better against *RB* and 25% better against *BB* in terms of average



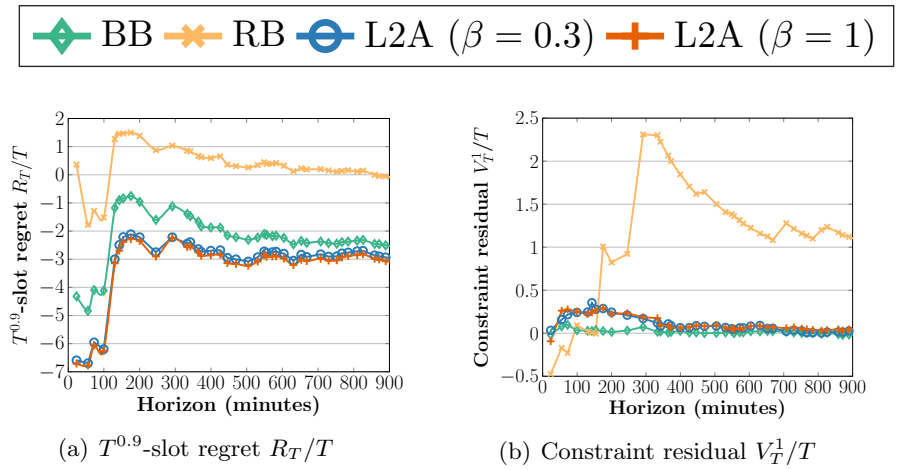
(a) Bitrate selection pedestrian user

(b) Bitrate selection markovian channel

(c) Buffer level pedestrian user

(d) Buffer level markovian channel

Figure 5.2: Sample paths for bitrate selection and buffer level



(a) $T^{0.9}$-slot regret R_T/T

(b) Constraint residual V_T^1/T

Figure 5.3: Convergence of regret and constraint residual

bitrate, while performing equally well, or even better (i.e. against *BB* in stability), in all other metrics. Thus *L2A* is *robust* against the channel fluctu-

ations and doesn't require any assumption on the channel rate distribution.

In Figure 5.2(a), a sample path for the channel rate and the bitrate selection for each method is presented for a randomly-selected pedestrian-mobility trace, while, Figure 5.2(c) depicts the evolution of the buffer for the same trace[2]. From these plots, we can argue that *L2A* learns the volatile channel distribution, in order to re-actively provide the highest bitrate (which is the optimization objective) and to pro-actively protect the buffer from under-flowing (which is one of the optimization constraints). While this 'adaptive behavior' of *L2A* may come only at a marginal cost in smoothness (i.e. bitrate distance between consecutive decisions), Figure 5.1(b) shows that L2A achieves on average only 3% less smoothness than the other methods; a trade-off that is aligned with common HAS optimization principles.

Similarly, we provide a bitrate selection sample path in Figure 5.2(b), for a randomly-selected markovian trace and its corresponding buffer evolution in Figure 5.2(d). Here, we observe: (i) that in terms of matching the bitrate to the channel rate at each decision epoch, *L2A* presents a more efficient channel utilization, (ii) a slightly unstable behavior for *BB*, especially at the beginning of the session and (iii) some stall events occurring for *RB*. *L2A* consistently manages to offer high bitrate, stable and uninterrupted streaming; even in the most demanding network scenarios.

To further investigate the *robustness* property of our method, we have synthesized an additional network profile, where we have concatenated *car* traces to extend the streaming session duration (horizon) in order to simulate longer, yet realistic scenarios. For these concatenated *car* traces, Figure 5.3(a) presents the regret rate R_T/T against the K-Slot benchmark of Section 5.3, for $K = T^{0.9}$. Here *L2A*, for both studied values of β $(0.3, 1)$, achieves better regret than any other method, significantly improving on the K-Benchmark in any streaming horizon, a result that is anticipated from Theorem 1.

Regarding the constraint residual V_T^i $\forall i = 1, 2$, we examine in particular the case of underflow ($i = 1$), as stalls are the most significant factors that can affect the streaming experience. Potential buffer overflows ($i = 2$) can be easily tackled by simply inducing a short delay before requests, according to (5.1). In Figure 5.3(b) we present the constraint residual rate for the concatenated *car* traces. *L2A* manages to respect the underflow constraint on average, given that the constraint residual rate V_T^1/T converges to 0.

In order to investigate the merits of *L2A* beyond VOD, we repeated the same cycle of experiments for the case of *live streaming*, where now $B_{max} = 20$s. In industrial live streaming applications, such small buffer values are commonly used, given the strict delay requirements. We present our results in Table 5.3, which depicts that although *RB* achieves higher values in stability, it is not able to compete with the other methods in terms

[2]We note here that in Figure 5.2, while some markers have been omitted for clarity, the lines remain an accurate representation of the results.

of average bitrate. On the contrary, *L2A* manages to provide up to 30% higher live streaming bitrate when compared to *RB* and equivalent bitrate to *BB*, while its switch-restricted instance shows up to 40% improvement in stability when compared to *BB*; in all considered scenarios. Online learning methods have – by design – less dependency on the instantaneous buffer length, when compared to buffer-based methods. They are also more reactive to throughput fluctuations, unlike throughput-based methods that are, normally, as efficient as their throughput estimation module.

5.6 Conclusions

In this work we present *Learn2Adapt* (*L2A*), a novel rate adaptation algorithm for HAS, based on online learning. Overall, our proposed method performs well over a wide spectrum of streaming scenarios, due to its design principle; its ability to learn. It does so without requiring any parameter tuning, modifications according to application type or statistical assumptions for the channel. The *robustness* property of *L2A* allows it to be classified in the small set of adaptation algorithms for video streaming, that mitigate the main limitation of existing mobile HAS approaches; the dependence on statistical models for the unknowns. This is of significant relevance in the field of modern HAS, where OTT video service providers are continuously expanding their services to include more diverse user classes, network scenarios and streaming applications.

Chapter 6

Video streaming traffic characterization

6.1 Introduction

Adaptive video streaming currently sustains the largest fraction of all global IP-traffic. Especially for mobile networks, HAS traffic is expected to generate 8 out of 10 packets that traverse the mobile networks [5] by 2022. Nonetheless, systematic studies of this traffic are rare in multimedia research. A significant part of the total research effort on HAS focuses on techniques and optimizations that improve overall QoE, such as advancing encoding technologies, optimizing network functions or proposing novel rate adaptation solutions. All these disciplines have a common characteristic; they either generate traffic that needs to be analysed in order to invoke its correlation with user QoE, or they rely on a-priori understanding of this traffic in order to infer realistic models and thus lead to efficient network management solutions. Therefore data-sets are a valuable resource to analyze, model and optimize network traffic and are more valuable than ever for communication research and industry practices.

Interestingly, HAS traffic has fundamental characteristics, which can be exploited for more efficient network resource management. For instance, schedulers with knowledge of video key parameters such as video rate and buffer level can adjust their priority weights and improve the overall QoE in the network. As mobile video streaming is constantly gaining popularity, a trend that is expected to be amplified in the following years, mobile network operators are witnessing an unprecedented increase in the number of concurrent HAS sessions that exist in their networks, at any given time. But, QoE can vary significantly as networks go through different utilization phases, especially in constrained networks such as the wireless last mile. To properly provision their networks and provide clients with the best possible service, operators need to understand traffic characteristics and how the users

QoE may vary as data flows compete for bandwidth. There is, therefore, an immediate requirement for network management solutions, designed to characterize streaming traffic and to identify early signs of insufficient QoE, such as stalls.

To this end, accurate knowledge of video streaming application-layer parameters is very useful for mobile operators in order to monitor and improve their network performance. Consequently, network optimization and traffic management need a method to provide video parameters to the network-layer in real time. This is known as cross-layer signalling. However, due to the recent wide use of data end-to-end encryption in the delivery of video streaming, the application of traditional solutions based on Deep Packet Inspection (DPI) and processing has become challenging. An alternative option for the operators is to rely on passive traffic measurements at the transport-layer within their network [148, 154].

To facilitate the study and design of such approaches, in this chapter we provide HAS traffic measurements at the network, transport and application layer. In particular we provide a large data-set for YouTube's popular video streaming client on mobile devices, that consists of 374 hours of time-synchronous measurements from two controlled environments in Europe. Our data-set allows a detailed analysis of HAS traffic at Layer 3 and 4 [12], depending on various network and video-related factors. A good statistical understanding at the packet level is particularly relevant with the deployment of new transport protocols such as QUIC [38]. Since the majority of our packet logs capture QUIC traffic, our data help operators and vendors to customize their networks and telecommunication equipment respectively, for this new kind of traffic.

Additionally, in Section 6.5 we present a novel reliable traffic profiling system for HAS, which fundamentally differs from DPI and cross-layer signaling. Instead of extracting or receiving information from the higher protocol layers, traffic profiling only observes a packet flow from the downlink transmit queue of the serving base station (BS), edge-router or gateway. In particular, we observe information such as source/destination address, size and arrival time of IP packets inside the network. Based on this input, our system separates HAS video flows from non-HAS traffic and estimates the current state of the video client's play-back buffer. Such real-time classification systems can then be used for streaming-specific billing and traffic shaping or to custom-tailor admission control and resource allocation [155] in cellular networks.

Besides this use-case, our data-set can be immediately applied to analyze HAS traffic across multiple layers from the perspectives of packet management, QoE factor analysis [156] and HAS algorithm design.

6.1.1 Prior work

Rao et al. [157] study the network characteristics of the two most popular video streaming services, Netflix and YouTube, and show that the streaming strategies vary with the type of the application and with the container type. The authors also identify three different streaming strategies that produce characteristic ON-OFF traffic patterns and present an analytical model to study the potential impact of these streaming strategies on the aggregate traffic.

Similarly, Sengupta et al. [158] provide traces from Android video apps collected under different values of parameters, such as video length, connection strength and device mobility, for the purpose of mobile video app traffic pattern identification. Mansy et al. [159] study Netflix, YouTube, and Hulu, over the two dominant mobile platforms – iOS and Android. They infer detailed session behavior based on passively collected packet traces over a large set of experiments across the providers and network types. Although all of these data-sets expand their data-set beyond YouTube, none provides application level information. They are among the few public data-sets that include measurement of mobile clients, however, they merely provide network traces. Additionally, to the best of our knowledge, the data-set presented in this chapter is the only existing HAS traffic data-set that includes QUIC.

Several interesting studies on the characteristics of HAS traffic were published. Ameigeiras et al. [160] formulate a traffic generation model for a YouTube server, based on measurements from a computer and re-evaluate the model for mobile devices in [161]. As QoE models are becoming very valuable for network performance validation, Wasmer et al. [156] provide such a study, based on subjective tests for YouTube. Their study introduces a network traffic model for the YouTube control mechanism and analyzes YouTube's operation from an end-user QoE perspective. The QoE of YouTube traffic is also studied by Liu et al. [162], using subjective tests and with an experimental setup similar to ours. Focusing on the redundant traffic and greediness of YouTube, Sieber et al. provide a very interesting study in [163].

The main idea of using passive traffic measurements for recognizing statistical patterns of video traffic has already been studied in several works, as for example in [154, 164–171]. Besides Dimopoulos et al. [165] who focus on the analysis and understanding of the obtained measurement data, the rest of the studies rely on machine learning techniques, a mathematical tool that has emerged in the research community as a prominent solution on the general topic of traffic classification. We encourage interested readers to refer to [164] and the references therein for a detailed comprehensive survey.

When it comes to video streaming, most of the recent works propose to distinguish the entire video session into different classes. For example, Orsolic et al. [166] propose a system that monitors application-level quality indicators and corresponding traffic traces to classify YouTube videos into three QoE

classes. Similarly, different levels of QoE are studied by Dimopoulos et al. [167] with the focus on stalling, average video rate and rate variations as the key influence factors. A causal analysis between QoE and QoS is presented by Katsarakis et al. [168] with features from application and network layer QoS metrics, while an approach to discriminate between audio and video HAS flows is proposed in [169]. Compared to these studies that try to classify each video into disjoint categories, our focus differs in the sense that we want to perform a more fine-grained analysis and estimate dynamic video parameters that may change throughout the streaming session.

Closer to our work is the methodology presented by Krishnamoorthi et al. [154] , where the target is to predict the class of the play-back buffer level during the video session by defining a set of buffer levels in seconds. Our focus is instead on the prediction of the buffer state that reflects more fundamental properties of HAS streaming. A similar traffic profiling system can be found in Tsilimantos et al. [172], where the system design, methodology and experimental validation for the estimation of the current state and bit-rate of a HAS session is presented and the classification of HAS states is based on simple heuristic algorithms. The work of presented in this chapter goes beyond [172] and proposes a solution that generalizes the applicability and robustness of traffic profiling by introducing and analyzing machine learning models for a wide set of challenging streaming scenarios.

6.1.2 Contributions

The contributions of this work are the following:

- We provide an extensive experimental design that enables researchers to study and to reproduce how a popular adaptive streaming client reacts to variable link conditions, or compare the performance of their own rate adaptation algorithms against YouTube's adaptive streaming policy.
- We specify an experimental setup that allows to design controlled experiments with automatic variation of network and streaming parameters during a video session.
- Besides measuring network statistics, we also provide streaming variables from YouTube's native Android application. This was possible by our recently introduced *Wrapper App* [173], which allows remote control and monitoring of YouTube's native Android application.
- We make our data-set publicly available[1] at [174] and we open-source all software tools to recreate it.
- All of these data are generated by YouTube's native Android application and most of the recorded logs refer to QUIC traffic, enabling the research community to study this new protocol for the first time.

[1] Accessible at: `http://qoecube.informatik.uni-wuerzburg.de/`

- we utilize our data-set to design a new traffic profiling system that classifies the flow type and play-back buffer state of HAS traffic at the IP layer.

We believe that publishing our data and tools will enable the research community to better understand how to model, manage and control adaptive streaming traffic.

The remainder of the chapter is structured as follows. Section 6.2 documents our experimental setup and design, including our *Wrapper App*. Our measurement procedures and measured variables are described in Section 6.3. Section 6.4 provides an overview of the currently available data. Section 6.5 introduces a novel traffic profiling system including the machine learning approaches for flow identification and buffer state classification along with experimental results based on our data-set. Section 6.6 provides ongoing applications and further ideas for using our data, while Section 6.7 summarizes the chapter.

6.2 Experimental design

In this section we describe the setup, scenarios, streaming content and location-specific differences for our experiments.

6.2.1 Setup

Our experimental setup consists of off-the-shelf consumer hardware, common tools for Linux and Android and customized measurement software for Android and Linux. For the latter, we are using our *Wrapper App* [173] to extract video information and streaming statistics from YouTube's Android player and control scripts to automatize the experiments.

The hardware setup is illustrated in Figure 6.1. A Linux computer (Kernel 3.16.0-71-lowlatency) controls two Android smartphones via Universal Serial BUS (USB) connections using the Android Debug Bridge (ADB) [175]. The control computer is connected to the Internet via a T1 line and operates as an Internet-gateway and Wireless Local Area Network (WLAN) access point for the smartphones. The smartphones perform video streaming via an IEEE 802.11g WLAN link at a carrier frequency of 2412 MHz. Due to the close distance between phones and access point, the average SINR was 23 dB, which provides the maximum physical layer rate of 54 Mbit/s.

The measurements are performed on the control computer and on the smartphones. On all the devices, network and transport-layer information is collected with tcpdump (version 4.7.4, libpcap version 1.7.4). On the smartphones, the native YouTube application (version 12.32.60) for Android generated HAS traffic over UDP according to QUIC [38]. Although we consistently observed some TCP packets at the beginning of each session, all

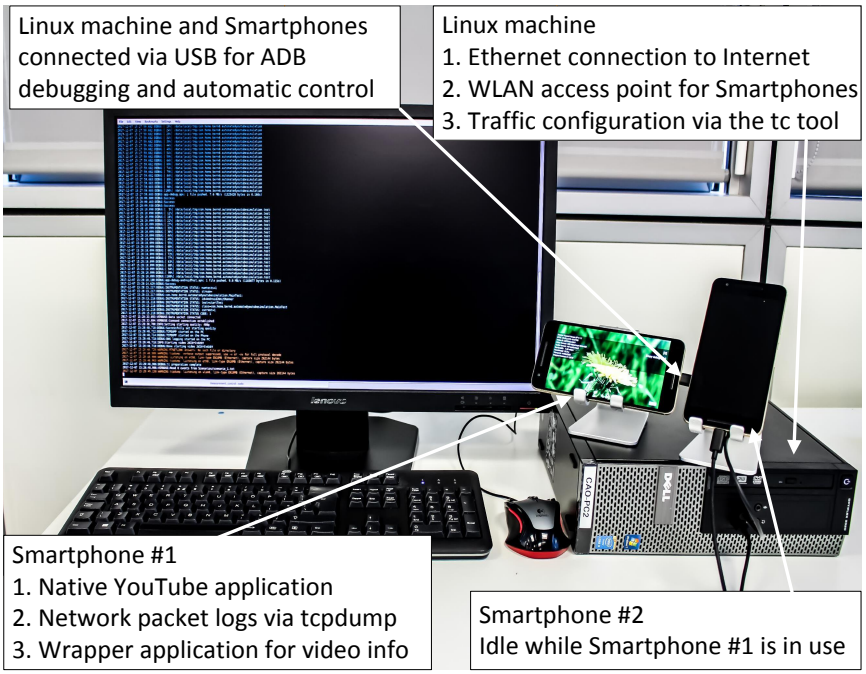


Figure 6.1: Hardware setup for regulated and automatic HAS traffic measurements

video streams were transported via UDP/QUIC. We assume that the TCP connections were used for tracking.

The setup allows to control the throughput, IP packet delay and Packet Error Rate (PER) that are experienced by the smartphones. This *traffic control* is done at the control computer via Python scripts that adjust a Token Bucket Filter (TBF) in the Linux kernel [176]. The scripts can automatically modify the above network parameters during an experiment at pre-determined or random points in time, which allows to emulate the dynamics of a mobile network in a reproducible manner.

In order to control and to extract application-layer information we designed the *Wrapper App* [173] – a measurement tool for the native YouTube Android application. In principle, our tool records application-layer information that is available through the *stats for nerds* feature of YouTube's Android application. As shown in Figure 6.2, this feature offers the values of some streaming-related parameters and the option to copy them to other applications via the clipboard. The *Wrapper App* accesses this information basically by executing touch-and-swipe events on the smartphone, a process which is remote-controlled by the measurement scripts with the help of ADB. In addition, the *Wrapper App* uses Android's *UI-automator* [177] to access the video progress bar.

These measurements are started and recorded periodically on the control computer. A measurement cycle is initialized by a control script, which starts

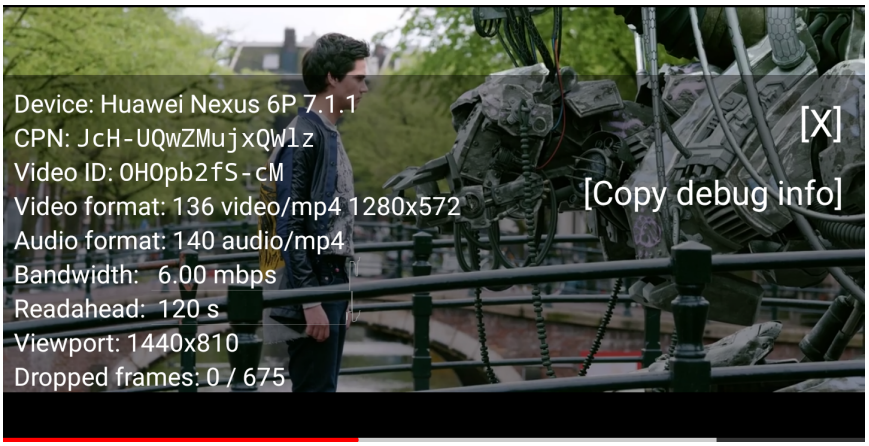


Figure 6.2: Screen-shot from the YouTube Android application during measurement with *stats for nerds*

the *Wrapper App*. This tool then launches the YouTube application and configures it according to the parameters in the scenario-specific configuration file (e.g., deactivating auto-play, choosing a fixed quality). The *Wrapper App* then starts the measurement by requesting the specified video.

It is important to note that the *Wrapper App* records only the mentioned application-layer measurements but no content (such as video, image data and audio).

6.2.2 Streaming scenarios

In order to cover a variety of representative streaming situations, we design and include in our experiments 8 different scenarios, as specified in Table 6.1 and listed below:

(s1) Medium quality (480p), no rate adaptation and no traffic configuration for the entire video.

(s2) High quality (720p), no rate adaptation, no traffic configuration for the entire video.

(s3) Quality Change (720p to 480p) at a random time in the interval $[120, 240]$s, no rate adaptation and no traffic configuration.

(s4) Adaptive quality, rate limitation at 500kbit/s starting at a random time in the interval $[120, 240]$s with a total duration of 150s.

(s5) Adaptive quality with constant rate limitation at 1024kbit/s for the entire video.

(s6) Adaptive quality based on DASH Industry Forum (DASH-IF) implementation guidelines [45, Table 5]. Besides the first step at 120s, all the next steps are applied every 40s. Rate, delay and PER traffic configurations are illustrated in Figure 6.3.

(s7) Adaptive quality, rate limitation at 120s by switching from 3Mbit/s to

Table 6.1: HAS scenarios

Scenario	Quality	Rate limit (kbit/s)
1	480p	Inf
2	720p	Inf
3	720p $\xrightarrow{t_0}$ 480p $t_0 \in [120, 240]\ s$	Inf
4	Auto	Inf $\xrightarrow{t_0}$ 500 $\xrightarrow{t_1}$ Inf $t_0 \in [120, 240]$s, $t_1 = t_0 + 150$
5	Auto	1024
6	Auto	(see Figure 6.3)
7	Auto	Inf $\xrightarrow{120\text{s}}$ 3000 $\underset{t_1}{\overset{t_0}{\rightleftarrows}}$ 100 $t_0 = 160 + 85n, n \in \mathbb{N}$ $t_1 = 205 + 85n, n \in \mathbb{N}$
8	Auto	Inf $\underset{t_1}{\overset{t_0}{\rightleftarrows}}$ 100 $t_0 = \{120, 300\}$s, $t_1 = \{180, 380\}$s

100kbit/s and back to 3Mbit/s every 40s and 45s respectively until the end of the video.

(s8) Adaptive quality, rate limitation of 100kbit/s starting at 120s and 400s, with a duration of 60s in both cases.

During the experimental design we made sure that our scenarios depict at least some clear state transitions. For this reason, we decided to leave the first 120s of each scenario without rate limits, besides (s5), since we verified from our experimental results that for the selected video content, a first state transition from filling to steady state appears during this time in normal streaming conditions. It should be noted that after the video starts playing, we assume that the user does not interrupt the video play-back by pausing or skipping forwards or backwards. Even though these events can be detected, we neglect them for simplicity.

Specifically, scenarios (s1) and (s2) are chosen in order to allow the study of simple streaming cases with a single filling and steady state, for two quality levels with significant bit-rate difference. Then, (s3) is selected to it represents multiple transitions between streaming states throughout the video session. (s4) is a more challenging scenario due to the introduced buffer depleting state as a result of rate throttling. The randomness in (s3)-(s4) is introduced in order to decrease the correlation with the video encoding distribution. The main reason behind the choice of (s5) is to include state changes under a rate that is common for 3G networks.

In (s6) we emulate more complex streaming conditions where the possibility of unclear buffer states is higher. Figure 6.3 shows the traffic configuration

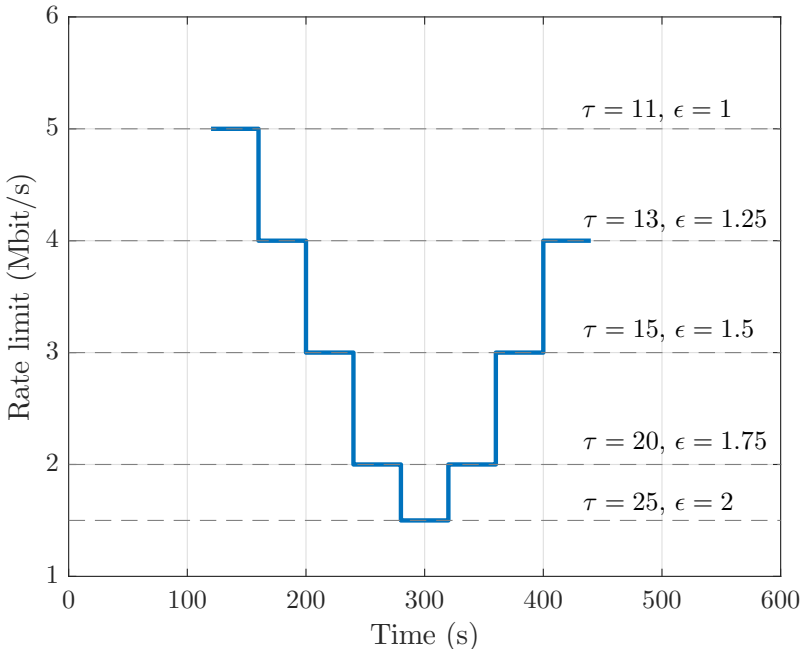


Figure 6.3: Traffic configuration of rate (Mbit/s), delay τ (ms) and PER ϵ (%) for (s6) according to DASH-IF guidelines

of this scenario, where a high-low-high rate profile is used. The indicated values of delay τ refer to transmission delay at the access point and should not be confused with the round trip time (RTT), which in our experiments had a mean value of 30ms. Finally, (s7)-(s8) are added to cover poor connection cases where streaming is not supported even with the lowest available video rate.

6.2.3 Streaming content

In order to provide data for representative videos, we chose 3 clips whose average encoding rates correspond to the average rates in [178, Table 4]. Table 6.2 provides the average μ and the standard deviation σ of the encoding rate for each streamed quality of the selected videos.

These videos represent different types of content. The movie Tears of Steel (*TOS*), with YouTube video ID is "OHOpb2fS-cM" [179], contains a mixture of computer generated and natural images in high motion. The clip is commonly used for testing video codecs and recommended in DASH-IF's specification [45, Section 2.1]. As a second clip we chose a nature documentary (*Nature*) [180] with YouTube video ID "2d1VrCvdzbY". This video consists of complex natural scenes (e.g., clouds, trees and water surfaces). Finally, we chose a talk-show (*TalkShow*) [181] as a low-motion clip with YouTube video ID "N2sCbtodGMI".

None of these videos is monetized, which prevents interruptions through

Table 6.2: Streaming content and bit-rate statistics (mean μ and standard deviation σ)

Video	**Duration (s)**		**Bit-rate (kbit/s)**					
			144p	240p	360p	480p	720p	1080p
TOS	734	μ	107	240	346	715	1347	2426
		σ	35	78	175	346	641	1152
Nature	561	μ	108	242	398	792	1566	3009
		σ	48	109	231	424	794	1381
TalkShow	559	μ	52	102	282	600	1156	2324
		σ	29	54	137	267	492	919

Table 6.3: Mean μ and standard deviation σ for 10000 measurements of RTT and 1954 measurements of throughput

Location	**RTT (ms)**		**Throughput (Mbit/s)**	
	μ	σ	μ	σ
Paris	18.64	24.17	9.07	2.17
Würzburg	25.19	34.41	13.01	4.63

advertisements. *TOS*, *TalkShow* and *Nature* are encoded at 24, 25 and 30 fps, respectively and are all available at the horizontal resolutions {144p–1080p}. Each of these qualities is available in two representations. First, as a H.264-encoded stream of DASH segments in an MP4 container and, second, as a VP9-encoded stream in a WebM container. YouTube's streaming application automatically chooses the format and indicates the chosen representation by an 'itag' field that is available in our dataset. The 'itag' index points to a specific field in the DASH manifest file, from which the content URL, format, URL, peak bitrate and other meta-data can be extracted according to the DASH specification [3].

6.2.4 Location-specific differences

The measurements were performed at two locations in Europe, i.e. Paris in France and Würzburg in Germany. While the main setup, procedures and scenarios were identical, two relevant differences between the setups were inevitable.

First, the performance of the T1 uplink to the Internet necessarily differed between Würzburg and Paris. While Würzburg's uplink provided 30% higher average throughput, it also came with a 26% higher average Round Trip Time (RTT). Our measurement results for those links are summarized in Table 6.3.

The second difference was the used smartphone model. In Paris, two identical phones of the type Huawei Nexus 6P (Model H1512, baseband

version: angler-03.78) were used alternately. Switching between these phones was necessary since, despite line power and a deactivated display, the measurements depleted the battery of the active phone faster than it could recharge. Using two phones allowed the inactive phone to recharge in order to provide continuous experiments. In Würzburg, the smartphone Google Pixel XL (Model marlin, baseband version: 8996-012511-1611190200) was used. This device did not suffer from battery depletion, making multiple phones unnecessary at this location. Note that both locations used the same version of the Android operation system, i.e., version 7.1.1 with the security patch from December 5$^{\text{th}}$, 2016.

Although we observed no significant influence of these location-specific differences, the reader should be aware that their effect was not systematically studied.

6.3 Data collection

In this section, the measurement procedures at the network, transport and application layer as well as the labeling process, are described in detail.

6.3.1 Network and transport-layer information

On the control computer and on the Android phones, we use the traffic capturing tool tcpdump to record TCP, UDP and IP information. Contrary to packet sniffing, we are using tcpdump not to record payload but to log meta data from the IP and UDP/TCP headers. For each packet that passes the TCP/IP stack, we configure tcpdump to produce a new record in a log file that contains a timestamp and the fields "*tos*: type of service, *ttl*: time to live, *id*: IP ID, *offset*: fragment offset, *flags*: fragmented diagram, *proto*: protocol type, *length*: packet length" and "source IP > destination IP: protocol, packet length". Packet length is given in bytes and the timestamp is in Unix epoch time (i.e. seconds after 00:00 UTC on 1 January 1970). Table 6.4 shows the most relevant parameters.

Note that we apply no filter to tcpdump, which means that we are logging all IP packets in both directions (ingoing as well as outgoing). Although all packet types are logged, YouTube's Android client employed QUIC for streaming. Thus, most of the recorded traces refer to UDP packets.

6.3.2 Application-layer information

We measure information at the application layer with two methods. First, we use our YouTube *Wrapper App* to extract information from YouTube's Android application itself. As described in Section 6.2.1, our tool transfers information from YouTube's statistic module and the video progress bar to the control computer. Video progress is recorded every 0.5 s in the file

Table 6.4: Selection of recorded parameters relevant to HAS

Layer	File identifier	Parameter
Network	TCPdump	packet arrival time (s)
	TCPdump	packet source IP address
	TCPdump	packet source port number
	TCPdump	packet destination IP address
	TCPdump	packet destination port number
Transport	TCPdump	packet payload size (Bytes)
	TCPdump	packet transport protocol
Application	DNS	query source IP address
	DNS	query source port number
	DNS	query destination IP address
	DNS	query destination port number
	Statistics	"bh": buffer level (ms)
	Statistics	video ID
	Statistics	"fmt": video quality (itag)
	Statistics	"afmt": audio quality (itag)
	Statistics	"bwe": bandwidth estim. (bit/s)
	Video_progress	elapsed video time (s)
	Event_log	network configuration events
	Event_log	Video play-out events

Video_progress, while the statistics are also extracted twice per second but stored in a file named *Statistics*. For each experiment, the *Wrapper App* logs its control events in the *event_log* file. A selection of the recorded variables is summarized in Table 6.4.

Our second method is to record information from Domain Name System (DNS) queries that are initiated by the YouTube application. These data are recorded by tcpdump and saved in the file *DNS*. This information allows to measure the initial delay by calculating the time difference between the playback start and the DNS initial request.

In order to illustrate some of the data that were recorded from the YouTube application during a streaming session, Figure 6.4 plots the level of the play-out buffer in seconds (left y-axis and both 'itag' lines) and the result of the bandwidth estimation (bwe) in Mbit/s over the session time. We observe that the adaptive streaming client initially chooses a quality of 720p (itag 136), which maintains the buffer level at around 120 s. After throughput throttling ($t = 152$ s), the play-out buffer leaves the steady state and starts depleting. YouTube's bandwidth estimation adjusts to this buffer depletion at $t = 208$ s, leading to the selection of the 144p quality (itag 160)

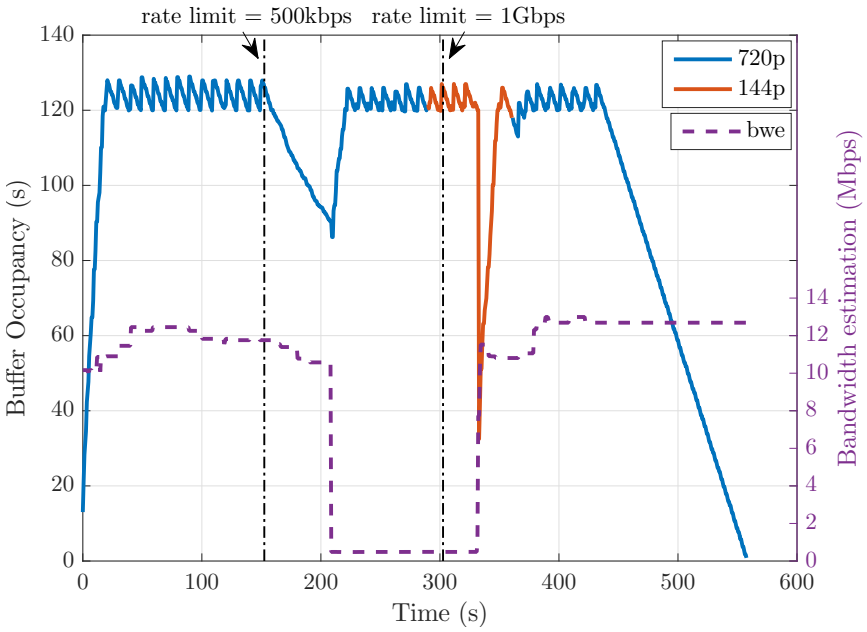


Figure 6.4: Example of data from *Statistics* and *Event_log* files, for video [181], under experiment scenario (s4).

at $t = 290\,\text{s}$. This drastic decrease in video quality allows to quickly fill up the play-out buffer and, thus, avoids stalls. After our control script has removed the throughput limit at $t = 303\,\text{s}$, YouTube's bandwidth estimation recovers at $t = 336\,\text{s}$ and the client requests 720p-segments again.

6.3.3 Buffer state labels

Figure 6.5 shows an example of a typical HAS session, as measured using our experimental setup and manually labeled according to our buffer state class definitions. The top figure shows the play-back buffer level in seconds, directly extracted from client's streaming application, while the bottom figure displays the accumulated data over time for the packet payload of streaming data, as recorded in our network traces. From this example, we can observe the three characteristic states of a HAS session. Firstly, while the play-back buffer is not sufficiently full, there is an initial burst of data where a new segment is requested immediately after the complete download of the previous one, leading to a streaming rate much higher than the video bit-rate. We denote this period as *filling state*, since the client quickly fills the buffer to a certain level, equal to 120s in this example. Once this target is reached, the *steady state* takes place where the requested streaming rate matches the video bit-rate, keeping the buffer level stable. This is achieved by a segment request pattern that leads to short packet bursts of one or more segments, followed by idle data transmission periods. Furthermore, due to the dynamic

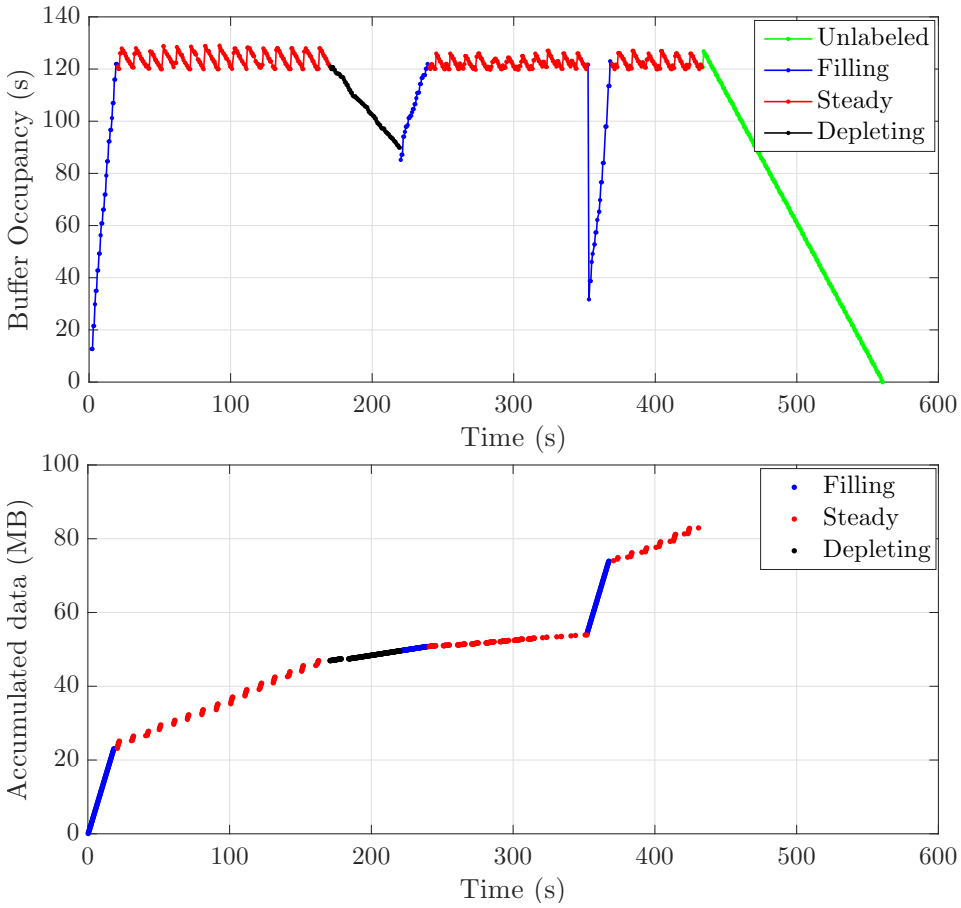


Figure 6.5: Example of labeled streaming video [181], using the setup in Section 6.2.1 under experimental scenario (s4).

wireless channel conditions and the presence of diverse bottlenecks in the video delivery system, it may happen that throughput is reduced below the video bit-rate during the streaming session. We impose this case in the example of Figure 6.5 by applying rate throttling in the interval $[170, 320]$s. At the beginning of this interval, the client tries to download data with the available throughput, but this is not enough to support the current video bit-rate and the buffer level inevitably decreases. We define this period as a *depleting state*. Then, in order to avoid a forthcoming video stall by letting the buffer to run empty, the HAS policy decides at some point to switch to a lower video quality, in this example at 220s, with an average bit-rate below the current throughput, leading to a second filling state and a subsequent steady state. After the end of the rate-throttling interval, higher throughput is again available and a new quality change leads to a third filling state, since the buffer is quickly filled with segments of higher quality. A last steady state takes place when the buffer target is reached again and then, finally,

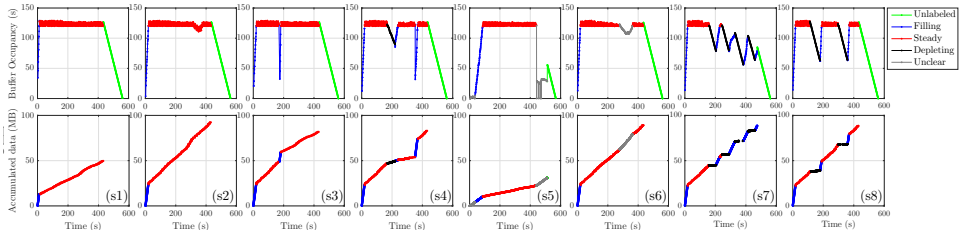


Figure 6.6: Examples of buffer level (top) and accumulated streaming data (bottom) per scenario for video [181], labeled according to Section 6.3.3.

after the entire video is transmitted, the session ends by playing-out the remaining bits from the buffer. We leave only this last part as unlabeled in the top figure, since there is no corresponding streaming data.

We labeled the buffer states manually with the help of a graphical user interface (GUI), that we developed for this reason, which provides plots as in Figure 6.5. The top figure shows the buffer level in seconds, as recorded in the *Statistics* log file. The bottom figure displays the accumulated data over time for the streaming data, as recorded in the *TCPdump* log file. By inspecting these plots, the GUI allows a user to:

- plot accumulated data separately per packet flow,
- specify time intervals for independent label assignment,
- associate a label with a previously defined interval.

The result of this process is a separate log file with timestamps and label values, that contains the string *Labels* in its name.

A representative example for each measured scenario is shown in Figure 6.6, where both buffer level and accumulated streaming data are labeled with the help of our GUI.

6.4 Data-set

Let us now provide an overview to the structure and the content of the data-set [182].

Using the setup and tools in Section 6.2 and the methodology in Section 6.3, the data were recorded in Paris, France and Würzburg, Germany, over the course of 5 months. The measurement campaign started on the 19$^{\text{th}}$ of September 2017 and ended on the the 23$^{\text{rd}}$ of February 2018.

During this campaign, 2201 experiments were performed. Each experiment is defined by its scenario index (s1 to s8), the video ID and the iteration index. Overall, 374 hours of streaming traffic were recorded, 25% of which are labeled. Table 6.5 provides the numbers of experiments per scenario and number of labeled experiments in parentheses.

The data are provided as text files in UTF-8 character encoding and csv-format. The directory structure is `./$Location/Scenario_$n/Vid_$v/Iteration_$m`, where Location = {'Paris', 'Wurzburg'}, $n = \{1, 2, \ldots, 8\}$,

Table 6.5: Total number of experiments and number of labeled experiments in parentheses

Scenario	**Video**			
	Nature	TalkShow	TOS	All videos
s1	90 (21)	102 (33)	109 (32)	301 (86)
s2	86 (17)	97 (27)	108 (31)	291 (75)
s3	80 (18)	94 (27)	108 (32)	282 (77)
s4	90 (25)	103 (35)	119 (41)	312 (101)
s5	63 (10)	82 (19)	102 (31)	247 (60)
s6	86 (32)	107 (40)	120 (43)	313 (115)
s7	81 (11)	86 (6)	87 (6)	254 (23)
s8	69 (7)	65 (3)	67 (4)	201 (14)
Total number	645 (141)	736 (190)	820 (220)	2201 (551)
Total time (h)	132 (29)	114 (30)	128 (34)	374 (93)

video ID v according to Section 6.2.3 and $m = \{1, 2, \ldots, M\}$ with iteration index M according to Table 6.5. For example, the directory `./Paris/Scenario_1/Vid_2d1VrCvdzbY/Iteration_1` contains the data of the first iteration of the *Nature* video according to scenario (s1) in Paris. Each directory `Iteration_$m` contains the csv-files, named according to the type of data {'Labels, 'DNS', 'TCPdump', 'event_log', 'Statistics', 'Video_progress'}, as described in Section 6.3. Some file names also indicate the device {'PC', 'Phone'}, as these measurements are simultaneously performed on the control computer ('PC') and on the smartphone. Note that we, redundantly, add \$Location, $\$n$, $\$v$ and $\$m$ to the file names for simplicity and organization.

Within each file, every line starts with a Unix timestamp [183], that is provided by the respective measurement device {'PC', 'Phone'}. Although the devices were synchronized via Network Time Protocol (NTP), we provide the file called *time_log* that contains both time references at the moment of initiation of the measurement.

6.5 HAS flow and buffer state classification

Recently mobile network operators have started deploying traffic shaping solutions in an effort to manage network resources according to market trends. In particular, in November 2015, T-Mobile USA deployed *BingeOn* which offers an unlimited plan for video streaming while throttling the video bit-rate to approximately 1.5 Mbit/s over an averaging window of 60 seconds [184]. A similar solution became operational in Germany in April 2017 under the term *StreamOn*. Other network operators are investigating similar solutions, while network equipment vendors are working on real-time reporting solutions for QoE and are custom-tailoring resource allocation mechanisms to adaptive

video streaming [185].

Before a video flow can be throttled or scheduled with specific rules, its packets need to be identified. Once identified, it is beneficial to know the video bit-rate and play-back buffer state of that stream in real time. This information allows schedulers, traffic shaping and admission control schemes to minimize their impact on QoE or to even increase it by providing bit-rate guarantees when possible [185].

This demand for accurate application-layer information is a major problem in practice. Network optimization functions typically operate at Layer 2 and 3 of the ISO/OSI protocol stack [12], while application information is available at Layer 7. Currently, network operators solve this cross-layer signaling problem by a combination of explicit signaling or DPI. DPI aims to dissect Layer 7 traffic at Layer 2 or 3 and to extract the required parameters, while explicit signaling requires OTT media services to add specific flags or tags to their streaming data. Both methods are susceptible to spoofing, where a malicious user can inject the appropriate flags or tags in its own traffic, which may not necessarily be a video stream. By using an HTTP proxy, Molavi Kakhki et al. demonstrated a simple spoofing method for BingeOn in [186] only several months after this service became available. In addition to spoofing, the use of end-to-end encryption techniques such as Transport Layer Security (TLS) and Secure Sockets Layer (SSL) is a major issue for DPI. Since most major streaming OTTs have adopted TLS/SSL, network operators cannot directly apply DPI to such encrypted video flows. In order to access the application-layer information, they have to interrupt the end-to-end encryption, e.g., by terminating the end-point on the client side. Since the OTT has to accept such "man in the middle attacks", this approach weakens operational security, leads to certificate errors with more rigorous clients, and eventually opens the door for malicious use.

In this section we provide a way to tackle the cross-layer signaling problem only by observing the IP packet flows of HAS traffic. A packet flow is defined as a series of packets sharing the same source and destination IP addresses, source and destination IP ports and transport protocol. Without loss in generality, we assume that packets are generated with a typical TCP/IP stack, i.e., TCP or UDP is used on top of IP. Packet flows are also distinguished in different classes, each class indicating IP traffic that belongs to a specific application, e.g. web, file transfer, gaming, peer-to-peer (P2P), chat and media streaming. During video streaming, the source node is a video server and the destination node is a video client software running on a mobile device. HAS is considered according to standards [3, 10] and the communication between server and client utilizes at least one intermediate node, e.g. edge-router, gateway, base station (BS) or access point, close to the edge that forwards the packets in either direction.

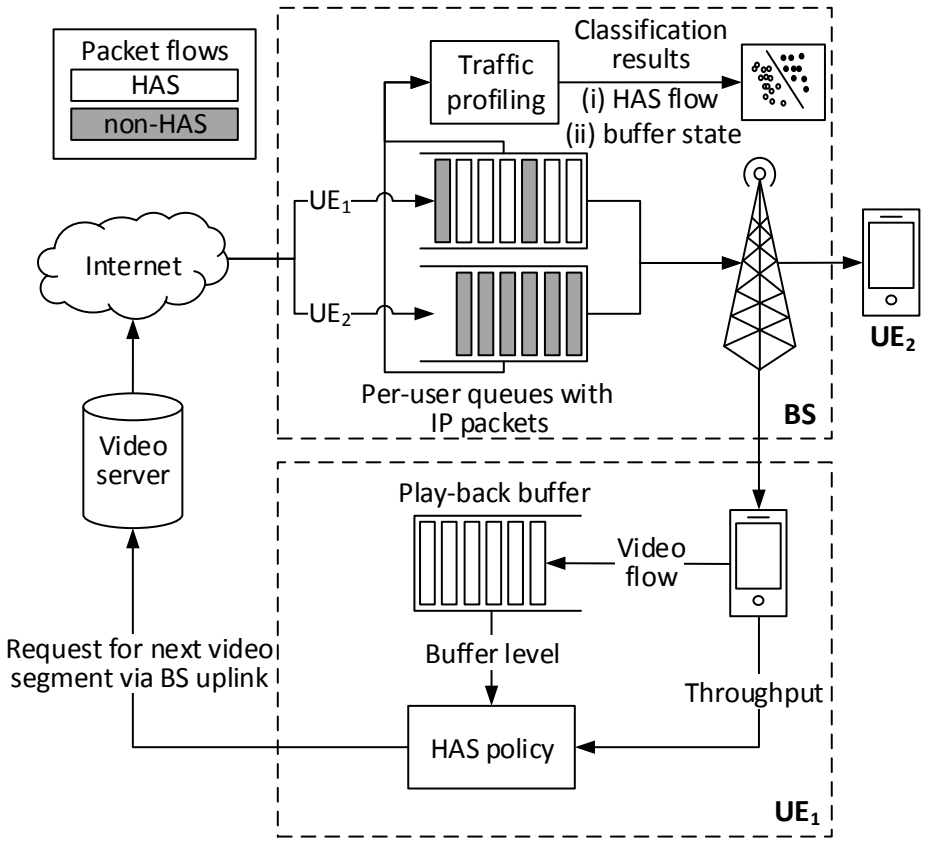


Figure 6.7: Example of traffic profiling at the BS with 2 users; UE_1 with HAS traffic and UE_2 with non-HAS traffic.

6.5.1 HAS traffic profiling

The proposed traffic profiling model adds a system module that monitors packet flows at the edge of a mobile network, i.e. at any edge-router, gateway, BS or access point. Figure 6.7 shows an application example with traffic profiling deployed at the BS monitoring the traffic of 2 user equipment (UE), where arriving packet flows are placed in user-specific queues and served by the BS scheduler.

At the core of traffic profiling, information directly observable at packet level is used to construct a set of features, which are used for the Machine Learning (ML) models. Our first classification problem is to accurately distinguish HAS flows from a set of packet flows with arbitrary traffic. Then, once a HAS packet flow is identified, we apply a second classification to detect the various buffer streaming states in real-time during the HAS session. In summary, the main functionality of the proposed traffic profiling system is as follows:

1. *Collection of packet information:* At the transport layer, observable

information at packet level is collected for each monitored packet flow, even for encrypted traffic.

2. *Construction of features:* For each flow, features are calculated based on the collected packet information and used to build ML models that recognize statistical properties of the flow. *For more information on how we construct the features, we refer the reader to [148]*

3. *Detection of HAS flows:* An HAS flow is distinguished from other packet flows in real-time by plugging the constructed features into the trained classifiers.

4. *Buffer state estimation for HAS flows:* Different streaming states for each HAS flow are identified in real-time by classification models.

At this point it is worth mentioning that the accurate buffer state estimation allows to extract useful video related application layer parameters. For instance, the observed streaming rate during each steady state can serve as an accurate estimator of video bit-rate [172]. One can also think of extensions related to QoE by estimating average video rate and stalling events.

A summary of all the defined classes is shown in Table 6.6. Video flow detection is formulated as a binary classification problem in our model. Each individual packet flow is classified either as 'HAS' or 'non-HAS' traffic, which in our case represents any measured non-HAS traffic, i.e. file downloads and web browsing. Since we are mainly interested in video streaming, a study that attempts to distinguish a multitude of different applications with one class per application is out of the scope of this work.

The problem of buffer state classification is more demanding, since there are more than two classes involved, potentially varying at any new sampling interval inside the same HAS flow. In this case we define a set of four different classes, i.e. 'filling', 'steady', 'depleting' and 'unclear', where the first three have been explained in Section 6.3.3 and the last one is simply introduced to cover few remaining cases and mainly comprises instances where the buffer is not close to the target level, but the video bit-rate is almost equal to the available throughput, leading to a slowly varying buffer until a new rate is selected.

Based on our training data-set, i.e the data-set of Section 6.4, in [148] we have evaluated the following five different ML classifiers, all of them well known in the ML literature:

1. *Support vector machines* (SVM): finds the best hyperplane separating data points of different classes.

2. *k-nearest neighbors* (KNN): each sample is assigned to the most common class among its k nearest neighbors.

Table 6.6: Set of defined classes

Problem	Name	Description
Service type	*HAS*	HAS traffic
	non-HAS	non-HAS traffic (web, downloads)
Buffer state	*filling*	streaming rate is higher than video bit-rate, buffer is filling
	steady	streaming rate matches video bit-rate, buffer target level is reached
	depleting	streaming rate is lower than video bit-rate, buffer level decreases
	unclear	all other cases, e.g. streaming rate is close to video bit-rate but buffer target level is not reached

3. *Boosted trees*: uses AdaBoost [187] algorithm to emphasize previously mis-modeled training instances.

4. *Random Forest* (RF) [188]: builds many decision trees and assigns instances to the class that most trees agree on.

5. *RUSBoost trees*: a hybrid sampling/boosting ensemble method with RUSBoost algorithm [189] for skewed training data.

While a representative performance evaluation with interesting insights on the effect of the selected features, machine learning method, and link quality can be accessed in [148], in the following will restrict our analysis to the best performing classifier therein, namely RF. For the classifier evaluation, we perform k-fold cross-validation, a common approach to train and validate a data-set. This means that the training data-set is randomly split into k equal sized partitions. Then, each partition in turn is used as the validation data and the remaining $k-1$ are used for training. This process is repeated k times with each of the k subsets used exactly once as the validation data-set.

6.5.2 Training data-sets

To obtain a sufficiently large training data-set, we resort to the data-set described in Section 6.4 and published in [174]. In particular, as we are interested in creating a training data-set for buffer state classification, we used only the measurements therein that contained labeled information about buffer states.

For the purpose of video flow classification, we also conducted measurements of non-HAS traffic, by including 2 additional scenarios, i.e. file

Table 6.7: Number of samples per class in the training data-set

Class	filling	steady	depleting	unclear	**Total**
HAS	23296	82225	9722	24297	139540
non-HAS	–	–	–	–	8071

download and web browsing, and included them in our training data-set. In practice, this type of mixed traffic reflects the presence of multiple users sharing the network at the same time, as the current version of native YouTube application does not allow multitasking. However, such a limitation can easily change in the future, e.g. users simultaneously streaming a YouTube video and accessing the web. Note that YouTube's RED subscription-based service [190] already provides such an option, by allowing to stream in the background while the YouTube application is minimized.

Based on our experimental testbed, that is detailed in Section 6.2.1, downloads of generic data files of size 100 MB, 200MB and 512 MB are performed by accessing a host server, i.e. `http://ipv4.download.thinkbroadband.com` in our case, via the built-in web browser of the smartphone. For web browsing, we developed an automatic framework that navigates to a specified web page from a list of pages via the built-in web browser of the smartphone and then proceeds to the next page in the list after a random amount of time of $[5, 15]$s, emulating users browsing on the web. In order to generate traffic with no trivial data size, we selected a list of pages that host a significant number of photos besides plain text. We explicitly exclude web pages that host video elements and leave the problem of classifying different types of video for our future work. Similarly to the video experiments, before every experiment the cache memory is cleared in order to avoid any caching issues for subsequent measurements.

The process of labeling is trivial for the binary classification problem of HAS flow detection, as we simply need to assign the same label for all sampling periods of a packet flow. By isolating experiments with HAS traffic, it is easy to detect HAS packet flows, by checking the total packet size of each flow, as only streaming flows should have significant size. However, in our approach we also visually verify labels by comparing the data pattern of a flow to the pattern of the buffer level obtained by the application layer.

Table 6.7 summarizes the number of samples per class in our training data-set, collected for a sampling period of $T_s = 1$s and manually labeled according to Section Section 6.3.3. Since our target is to classify transmitted data per sampling period, the values in Table 6.7 count only non-empty samples, i.e. seconds with one or more packets. This filtering allows us to reduce the data-set size by keeping only meaningful entries. As previously explained, this data-set is used for both training and testing by applying k-fold cross-validation.

Table 6.8: Confusion matrix for flow classification ($k = 10$)

True	**Predicted**	
(%)	HAS	non-HAS
HAS	100.0	0.0
non-HAS	0.2	99.8

Table 6.9: Confusion matrix for buffer state classification ($k = 10$)

True	**Predicted**			
(%)	filling	steady	depleting	unclear
filling	98.7	0.5	0.6	0.2
steady	0.1	99.7	0.1	0.1
depleting	1.4	0.9	97.6	0.1
unclear	0.2	0.5	0.0	99.3

6.5.3 Results

Overall accuracy is defined here as the ratio of correctly classified samples over the total number of samples. The performance evaluation is based on k-fold cross-validation, for $k = 10$. A confusion matrix is presented in Table 6.8, where we can observe that HAS traffic is almost always classified correctly, while the overall accuracy is mostly affected by *false positives*, i.e. samples of non-HAS traffic incorrectly classified as HAS traffic. However, the percentage of false positives is very low with a value less than 0.2%. As a general remark, one should keep in mind that the classification results are per sampling period. In practice, a packet flow cannot change class until its termination, since it either belongs to HAS or non-HAS traffic. This fact enables post-processing methods to perform a second step of flow classification based on the existence of a dominant class, where isolated samples with predicted class different from the dominant predicted class can be neglected.

Having confirmed the high accuracy of HAS flow classification, we proceed with studying the performance of buffer state classification. The respective confusion matrix is presented in Table 6.9, where again we witness very high accuracy for the buffer state classification task. We note that the classification error of RF, particularly for the case of falsely classifying depleting states as filling states, is a function of the number of trees, which is a key parameter for its implementation. A small number of trees (30) is used initially, as we want to keep low complexity and memory requirements, but in [148] we detail more on the impact of this parameter.

Last, although here we only presented classification results for RF, a performance evaluation of each ML method that we studied is available at [148]. Nonetheless, it is worth verifying how challenging is buffer state

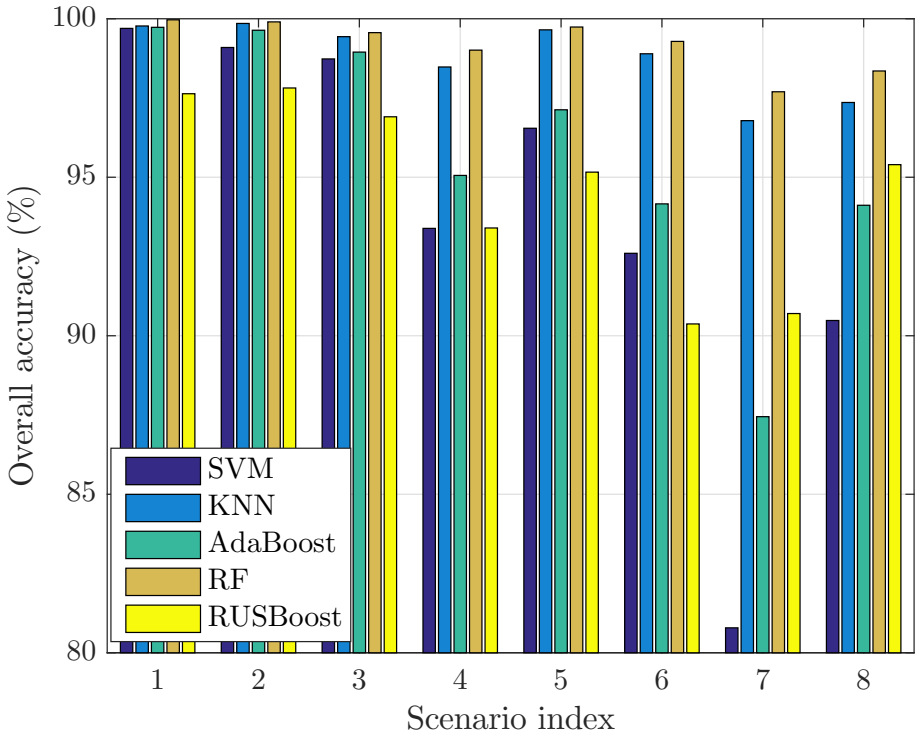


Figure 6.8: Overall accuracy per scenario for buffer state classification by 10-fold cross-validation.

classification for each designed scenario not only for RF but for each studied ML algorithm. Thus, Figure 6.8 shows the overall accuracy per experimental scenario. As expected, all studied algorithms perform better for (s1)-(s2), i.e. the simplest scenarios in our set involving a single filling and steady state. Their performance is slightly reduced for (s3), mainly due to the transition period that follows the manual rate change. (s4)-(s8) are clearly more challenging, since more buffer state changes are introduced. Nevertheless, RF manages to maintain an accuracy higher than 97.7%, for all scenarios.

6.6 Other applications

Our data-set provides time-synchronized measurements at the ISO/OSI [12] Layers 3, 4 and 7 in controlled scenarios. While these data may be used in various ways, let us now briefly point to specific applications for (i) designing adaptive streaming clients, (ii) analyzing streaming traffic at packet level and (ii) estimating streaming parameters and quality.

Our data-set allows to study and to reproduce how a popular adaptive streaming client reacts to variable link conditions. By mapping the scenario parameters from Table 6.1 to the measurements at Layer 7 (cp. Table 6.4) algorithm designers can analyze, for instance, the reaction time and video

quality of YouTube's client depending on throughput, PER and delay. Since our measurement tools are publicly available [173, 182], researchers can even reproduce our test-bed and compare the performance of their own rate adaptation algorithms against YouTube's adaptive streaming policy. In the same manner, our tcpdump-logs can be used to study and to improve the packet generation of adaptive streaming clients compared to YouTube.

Additionally, our data-set allows to estimate application-layer parameters and quality, based on packet flows. In [172], we used a previous version of this data-set to estimate video encoding rate only by observing the size and arrival time of the related IP packets. Such simple mechanisms already provide sufficient accuracy for adjusting resource allocation functions [155] and for estimating QoE [156]. Similarly, our data-set can be used to estimate further QoE factors such as initial buffering delay [191], rate adaptation frequency as well as stalling frequency and delay.

6.7 Summary

In this chapter we described our public [182] data-set for YouTube's mobile streaming client and the methods to reproduce it.

Our experimental setup allows to design controlled experiments with automatic variation of network and streaming parameters during a video session. Besides measuring network statistics, streaming variables are extracted from YouTube's native Android application. Our procedures support the test cases of the DASH Industry Forum [45] and provide state labels for YouTube's adaptive streaming logic.

At the moment, our data-set offers 374 hours of synchronous measurements at the network, transport and application layer in two controlled environments. All of these data are generated by YouTube's native Android application and most of the recorded logs refer to QUIC and UDP traffic.

Our data-set allows a detailed analysis of adaptive streaming traffic at Layer 3 and 4, depending on various network and video-related factors. A good statistical understanding at the packet level is particularly relevant with the deployment of new transport protocols such as QUIC. Since the majority of our packet logs capture QUIC traffic, our data help operators and vendors to customize their networks for this new kind of traffic. One example of such customization is service classification based on packet arrival patterns [148], which we developed based on the presented data-set and describe in detail in Section 6.5. This new traffic profiling system classifies flows and buffer states of HAS traffic, based on machine learning. The core of our approach is a classifier that separates HAS from non-HAS traffic as well as 4 states of the HAS play-back buffer.

Chapter 7

Conclusions

In this section, we summarize the work presented in this dissertation and outline several research directions that are currently arising in the field of multimedia delivery and which we identify as worth of future research exploration in the following years.

7.1 Summary

In Chapter 1 we provided a historical background and an introduction to the basics of video streaming and the DASH standard in particular, along all related technologies and a brief discussion concerning the relevant factors that influence video streaming QoE.

In Chapter 2 we provided a literature review of the state-of-the-art in HAS design, where we made a two level classification of existing HAS methods, initially according to location of the rate adaptation logic in DASH architecture (sever-side or client-side) and then according to the dynamic used in the adaptation logic (throughput-based, buffer-based and hybrid).

In Chapter 3, we evaluated the performance of five state-of-the-art adaptive streaming algorithms and made a per class comparison, based on real network traces for multiple throughput profiles. Additionally, we evaluated the impact of the buffer size to see how each strategy behaves for smaller and larger buffers, as different services (VoD or live streaming) may target different buffer levels. Our conclusion is that buffer-based approaches perform better in terms of average rate, when compared with the other evaluated classes of algorithms, yet they may present diminished rate adaptation stability, especially for small buffers, common in live streaming services. This work provides first guidelines to designers and operators of HAS algorithms for the right algorithmic approach according to expected network conditions and service requirements. Based on the results of our performance evaluation in mobile networks, we have identified two critical limitations of most existing rate adaptation methods. In particular these issues concern (i) the singular

focus on network and application-layer inputs into the adaptation process, while disregarding cross-layer dynamics and (ii) the dependence on statistical model for the unknowns. Motivated to address these two issues, we directed our research towards designing novel adaptation algorithms presented in Chapter 4 and Chapter 5, respectively.

In Chapter 4, we extended a classic rate adaptation algorithm by a Bayesian coverage estimator, incorporating inferred location information into the rate adaptation process. The resulting context-aware buffer-based rate adaptation algorithm, called *Indoors-Outdoors Buffer-Based Adaptation (IOBBA)*, is provided with context information on a user's location in relation to a building, inferred by the fusion of cellular and geo-localization sensor readings. The coverage estimation requires no knowledge of user location and makes no use of coverage or map data. The required input is collected at run-time, and the rate adaptation mobile client adjust its decision strategy accordingly. Video rate is aggressively reduced if the user is estimated to enter an area of poor signal coverage (e.g., indoors), and video rate increases are made more conservatively, until an area of high coverage is detected (e.g., outdoors). Our experiments in a typical downtown office building show substantial improvements in smoothness and stability, compared to a conventional rate adaptation algorithm. We have made a video, illustrating the measurement scenario and the QoE gain for adaptive streaming, available online[1], which clearly demonstrates that making adaptive streaming aware of the user's current coverage situation can highly improve the QoE of mobile streaming.

In Chapter 5 we recasted the video streaming problem as an online optimization problem that we solved with online learning. The resulting solution, *Learn2Adapt*, is a novel rate adaptation algorithm based on the OCO theory. Overall, our proposed method performs well over a wide spectrum of streaming scenarios, due to its design principle; its ability to learn. Unlike several existing rate adaptation, it does so without requiring any parameter tuning, modifications according to application type or statistical assumptions for the channel. The *robustness* property of our method allows it to be classified in the small set of rate adaptation algorithms that mitigate the main limitation of existing HAS approaches, according to Chapter 3; the dependence on statistical models for the unknowns. This is of significant relevance in the field of modern HAS, where OTT video service providers are continuously expanding their services to include more diverse user classes, network scenarios and streaming applications.

In Chapter 6 we studied HAS, under a new perspective. In Chapter 3 to Chapter 5, our work has established that robust rate adaptation algorithms are critical for OTT service providers who are interested in optimizing user QoE, especially in the significantly more challenging mobile networks.

[1] Accessible at `https://youtu.be/qJibE2N37Yk`

Nonetheless, we have identified that the absence of updated streaming traffic is a standing and pressing issue in multimedia research. HAS traffic is crucial for understanding the streaming principle, under a network operator's point-of-view and can unlock resource management optimizations in many planes. To this end, in Chapter 6 we addressed this requirement, by providing a large data-set of cross-layer HAS traffic measurements. Precisely, we described our public data-set for YouTube's mobile streaming client and the methods to reproduce it. Our experimental setup allows to design controlled experiments with automatic variation of network and streaming parameters during a video session. Besides measuring network statistics, streaming variables are extracted from YouTube's native Android application. Our procedures support the test cases of the DASH Industry Forum [45] and provide state labels for YouTube's adaptive streaming logic. Our data-set offers 374 hours of synchronous regulated measurements at the network, transport and application layer in two locations in Europe (France and Germany). All of these data are generated by YouTube's native Android application and most of the recorded logs refer to QUIC and UDP traffic. We have published not only our data, but also all necessary tools to reproduce it, in order to enable the research community to better understand how to model, manage and control adaptive streaming traffic. Additionally, we explain how we use this data-set to analyze adaptive streaming traffic and to design traffic managing functions for cellular networks. In particular we introduce a new traffic profiling system that, classifies data flows and buffer states of HAS traffic, based on machine learning. The core of our approach is a classifier that separates HAS from non-HAS traffic as well as 4 states of the HAS play-back buffer.

7.2 Future research perspectives

Throughout this dissertation, several concrete developments have been proposed, towards optimizing mobile streaming QoE and additionally, towards understanding HAS traffic that currently represents more than have mobile data traffic [1]. Nonetheless, the constant evolution in telecommunications and multimedia technologies are perpetually changing the landscape of streaming services. As these changes in the industry develop and mature, we will see many new applications that require new streaming solutions to be developed. In the context of these next-generation services, there are many interesting and important emerging challenges, which will need to be addressed by the multimedia management community in the near future. In this section, the main challenges and the newly arising opportunities associated with them, are explored.

7.2.1 HAS over QUIC

An emerging research topic explores the kind of advantages that can be obtained when the transport layer is modified to better support HAS traffic. The recently proposed QUIC [39] by Google, is a new transport layer protocol based on UDP that can be implemented in the user space rather than the system kernel (which is the case for TCP), and can therefore be deployed and updated more easily than TCP. To guarantee reliability, QUIC has to implement a congestion control algorithm, like Google's BBR algorithm [40]. QUIC provides several interesting improvements compared to TCP, as for instance true multiplexing of HTTP/2 streams at the transport level; as opposed to standard TCP that still introduces HOL blocking when the packets of a certain stream are lost.

In Chapter 6 we detailed our HAS over QUIC data-set that would allow to study the merits of QUIC on HAS QoE. Initial studies by the team that developed QUIC, report a re-buffering rate reduction of YouTube playbacks by 18.0% for desktop users and 15.3% for mobile users [38], compared to standard TCP. Nevertheless, a formal analysis of adaptive streaming over QUIC has only been marginally investigated [192], and is still missing at the moment. The impact of QUIC on adaptive streaming and the network conditions where it outperforms standard TCP, along with the potential merits of developing a congestion control algorithm tailored on the adaptive streaming needs, remain unexplored.

7.2.2 Distributed video streaming

De-centralized solutions such as Peer-to-Peer (P2P) Streaming [193] pose attractive alternatives for delivering video over the Internet. In a P2P network, each peer (user), while downloading a video stream, is simultaneously also uploading that stream to other peers, thus contributing to the overall available bandwidth. According to the architecture of many P2P networks, if a peer wishes to view certain content, the P2P software contacts a "tracker server" for that content in order to obtain addresses of peers who distribute that content; it then contacts these peers to receive the feed. The tracker records the peer's address, so that it can be given to other peers who wish to view the same content. In effect, this creates an overlay network on top of the regular internet. There are two categories of P2P network: tree-based and mesh-based. On the one hand, tree-based systems organize peers in a tree structure and video is pushed from the root to subsequent levels of the tree until it reaches the leafs, a model that, although simple and easy to control, can be hindered by participant turnover [194]. On the other hand, in mesh-based systems, peers connect to a set of neighbouring peers who watch the same content, exchanging information about their data availability and retrieving data from neighbours. This model is much less affected by

turnover, since each client maintains a set of peers at any point in time. P2P streaming in the form of BitTorrent is still popular for video file sharing today.

The main advantages of P2P streaming are that it does not need support from Internet routers; thereby it is cost effective and simple to deploy, and that it has the ability to maximize the video quality delivered to individual peers in a scalable fashion, in spite of the heterogeneity and irregularity of their access link. The aggregate available resources in this approach physically grow with the user population and can potentially scale to any number of participating peers. However, providing video streaming services to a large number of heterogeneous peers still remains an open issue, as it creates challenges on effective system and network resource management. The merit of enabling adaptive streaming, via the DASH standard, for P2P streaming systems has only marginally been explored [195, 196]. Thus further research, motivated by the advantages and success of the P2P network model in media content distribution and the characteristics and benefits of adopting the DASH standard in a streaming system, can yield merit-able results. Jointly exploiting the cooperation property of P2P and the flexibility of DASH has the potential to facilitate high quality streaming to a large population of users.

7.2.3 Immersive video streaming

As head-mounted devices, are experiencing continuous development and are quickly becoming more accessible, the demand for omnidirectional and/or immersive videos is expected to grow in the next years. In the context of 360°or Virtual Reality (VR) streaming, the users have complete control over the selection of the the viewing position, known as the viewport. Due to the high bandwidth requirements of 360°videos, immersive streaming are often characterized by low-quality. In that direction, viewport-dependent solutions have been proposed [197], that are able to reduce the required bandwidth. The main principle of such view-aware solutions is distinguishing the portion of the video that is in the viewport and outside the viewport. In-viewport video is streamed in high-quality whereas the rest is streamed at a lower quality or not streamed at all.

Tiled-based adaptive streaming [198] has been recently proposed and represents an extremely interesting solution in HAS domain, as MPEG-DASH has recently standardized a new specification, called spatial representation description, to support tile-based streaming [199]. Additionally, HEVC offers the possibility to encode each picture as a set of independent smaller parts, the tiles. There the two major solutions to get the bandwidth down to realistic levels: (i) by creating many different versions of the panoramic picture, and streaming the one that best fits the viewport and (ii) by dividing the image into tiles, and only sending the tiles that are in view. The new

HTTP/2 protocol, with its new features as server push, stream prioritization and multiplexing, can represent a possible future research direction [200]. Currently the most important open question in regard to tiled-streaming, concerns the prediction of changes in the users' viewpoint and how these changes may affect QoE.

7.2.4 Personalized QoE

As briefly discussed in Chapter 1, most employed QoE models are developed to capture the behavior of the average user, and are therefore not personalized. Also, standard models do not consider the context in which the streaming session takes place. Typically online frameworks for HAS, such as the one presented in Chapter 5, incorporate network and application level information as mean measures of QoE in the feedback control. To capture the actual user experience, personalized QoE-aware streaming represents a possibility worth exploring. Personalized QoE models, which are representative of the specific, rather than aggregate, user behavior should not only involve application-level parameters, such as rate switches or stalls, but also sensor inputs and context [201, 202]. Current smartphones and tablets are already equipped with several sensors (e.g., GPS, light sensors etc.), and wearable devices will allow to obtain even more fine-grained information on the user status (e.g., heart level, eye-tracking etc.). Using both network, application and user-level parameters will allow to create real personalized QoE models.

In Section 6.5, we established that machine learning algorithms represent a good candidate for QoE inference. Nonetheless some open issues exist. Since performing such a computational intensive task directly on the users device is currently infeasible, to make personalized QoE a viable option for mobile streaming we need to rely on cloud offloading; which would require real-time delay restrictions of the QoE modeling task. Once this issue is resolved, a complex model that encompasses not only application-level but also user-level parameters could be used to estimate on-the-fly and online the real QoE of the specific user. The online feedback from the user can be both explicit, if the user can directly rate the viewing experience, and implicit, when the sensor information is used to estimate the personalized users QoE. This aspect opens up the possibility to control the service directly at the user level.

7.2.5 Adaptive streaming in 5G

With the advent of the next generation of cellular technology, i.e. 5G, the significantly increased bandwidth will expedite bandwidth-intensive multimedia applications, such as Ultra High Fidelity mobile streaming or mobile AR/VR applications.

Yet, another major current trend for next-generation telecommunication

networks, is to evolve towards enabling extremely low-latency streaming applications, such as real-time communications and ultra reliable monitoring, for instance in the context of tele-medicine.

Additionally, 5G will be able to support both traditional forms of communication and more unstructured ones, such as machine-to-machine communication. For instance, autonomous cars are becoming smart and connected and some already have streaming capabilities; such as in-car entertainment systems. As they can already stream both from a network infrastructure or from each other [203], it is evident that a shift occurs in the delivery architecture of these new applications.

Overall, adaptive video streaming will have a central role to enable these new dynamic services, especially when combined with new network paradigms as 5G and softwarized networks [204].

Thus streaming solutions have to be both flexible and autonomously reconfigurable, depending on the application and its requirements. Therefore, context-awareness, both applicable in the video client, as initially explored in Chapter 4, and in the network will also be crucial for optimized streaming in such a dynamic ecosystem.

Bibliography

[1] Cisco Visual Networking Index, "Global mobile data traffic forecast update, 2017–2022," *white paper*, Feb. 2019.

[2] Sandvine, "Global Internet Phenomena," *Report*, Jun. 2019.

[3] ISO/IEC, "Dynamic adaptive streaming over HTTP (DASH)," *International Standard 23009-1:2014*, May 2014.

[4] T. Karagkioules, C. Concolato, D. Tsilimantos, and S. Valentin, "A Comparative Case Study of HTTP Adaptive Streaming Algorithms in Mobile Networks," in *Proc. ACM NOSSDAV*, Jun. 2017, pp. 1–6.

[5] Cisco Visual Networking Index, "Forecast and Trends, 2017–2022," *white paper*, Feb. 2019.

[6] ITU-T G.984.1, "Gigabit-capable passive optical networks (GPON): General characteristics," *Technical report*, Mar. 2008.

[7] 3GPP TR25.913, "Requirements for Evolved UTRA (E-UTRA) and Evolved UTRAN (E-UTRAN)," *Technical report*, Jan. 2015.

[8] Microsoft Silverlight Smooth Streaming. [Online]. Available: https://www.microsoft.com/silverlight/smoothstreaming/

[9] HTTP Dynamic Streaming (HDS). [Online]. Available: https://www.adobe.com/products/hds-dynamic-streaming.html

[10] R. Pantos and W. May, "HTTP live streaming," *IETF, Informational Internet-Draft 2582*, Sep. 2011.

[11] T. Stockhammer, "Dynamic Adaptive Streaming over HTTP –: Standards and Design Principles," in *Proceedings of the Second Annual ACM Conference on Multimedia Systems*, ser. MMSys '11, 2011, pp. 133–144.

[12] ISO/IEC, "Open Systems Interconnection (OSI)- Basic Reference Model," *International Standard 7498-1:1994*, Nov. 1994.

[13] S. Essaid, R. El-Azouzi, and E.-H. Bouyakhf, *Cross-layered QoS Framework for Next Generation Wireless Networks.* Editions Universitaires Europeennes, 2011.

[14] Encoding.com, "Global Media Format Report, 2017," *report*, Feb. 2017.

[15] T. Wiegand, G. J. Sullivan, G. Bjontegaard, and A. Luthra, "Overview of the H.264/AVC video coding standard," *IEEE Transactions on Circuits and Systems for Video Technology*, vol. 13, no. 7, pp. 560–576, July 2003.

[16] S. Varma, *Internet Congestion Control*, 1st ed. San Francisco, CA, USA: Morgan Kaufmann Publishers Inc., 2015.

[17] S. Rimac-Drlje, O. Nemcic, and M. Vranjes, "Scalable Video Coding extension of the H.264/AVC standard," in *2008 50th International Symposium ELMAR*, vol. 1, Sep. 2008, pp. 9–12.

[18] G. J. Sullivan, J. Ohm, W. Han, and T. Wiegand, "Overview of the High Efficiency Video Coding (HEVC) Standard," *IEEE Transactions on Circuits and Systems for Video Technology*, vol. 22, no. 12, pp. 1649–1668, Dec 2012.

[19] C. Concolato, J. Le Feuvre, F. Denoual, F. Maz, E. Nassor, N. Ouedraogo, and J. Taquet, "Adaptive Streaming of HEVC Tiled Videos Using MPEG-DASH," *IEEE Transactions on Circuits and Systems for Video Technology*, vol. 28, no. 8, pp. 1981–1992, Aug 2018.

[20] J. M. Boyce, Y. Ye, J. Chen, and A. K. Ramasubramonian, "Overview of SHVC: Scalable Extensions of the High Efficiency Video Coding Standard," *IEEE Transactions on Circuits and Systems for Video Technology*, vol. 26, no. 1, pp. 20–34, Jan 2016.

[21] J. D. Cock, A. Mavlankar, A. Moorthy, and A. Aaron, "A large-scale video codec comparison of x264, x265 and libvpx for practical VOD applications," in *In Proc. SPIE Optical Engineering and Applicaitons*, vol. 9971, 2016.

[22] L. Guo, J. De Cock, and A. Aaron, "Compression Performance Comparison of x264, x265, libvpx and aomenc for On-Demand Adaptive Streaming Applications," in *2018 Picture Coding Symposium (PCS)*, June 2018, pp. 26–30.

[23] A. Zabrovskiy, C. Feldmann, and C. Timmerer, "A Practical Evaluation of Video Codecs for Large-Scale HTTP Adaptive Streaming Services," in *2018 25th IEEE International Conference on Image Processing (ICIP)*, Oct 2018, pp. 998–1002.

[24] C. Timmerer, "MPEG Column: 122Nd MPEG Meeting in San Diego, CA, USA," *SIGMultimedia Rec.*, vol. 10, no. 2, pp. 4:4–4:4, Aug. 2018.

[25] R. R. Stewart, "Stream Control Transmission Protocol," RFC 4960, Sep. 2007. [Online]. Available: https://rfc-editor.org/rfc/rfc4960.txt

[26] S. Floyd, M. J. Handley, and E. Kohler, "Datagram Congestion Control Protocol (DCCP)," RFC 4340, Mar. 2006. [Online]. Available: https://rfc-editor.org/rfc/rfc4340.txt

[27] K. B. Egevang and P. Francis, "The IP Network Address Translator (NAT)," RFC 1631, May 1994. [Online]. Available: https://rfc-editor.org/rfc/rfc1631.txt

[28] T. L. Hain, "Architectural Implications of NAT," RFC 2993, Nov. 2000. [Online]. Available: https://rfc-editor.org/rfc/rfc2993.txt

[29] D. B. D. A. Ph.D., E. B. Davies, and E. B. Davies, "Reflections on Internet Transparency," RFC 4924, Jul. 2007. [Online]. Available: https://rfc-editor.org/rfc/rfc4924.txt

[30] H. Schulzrinne, S. L. Casner, R. Frederick, and V. Jacobson, "RTP: A Transport Protocol for Real-Time Applications," RFC 3550, Jul. 2003. [Online]. Available: https://rfc-editor.org/rfc/rfc3550.txt

[31] C. Holmberg, S. Hakansson, and G. Eriksson, "Web Real-Time Communication Use Cases and Requirements," RFC 7478, Mar. 2015. [Online]. Available: https://rfc-editor.org/rfc/rfc7478.txt

[32] C. Perkins and V. Singh, "Multimedia Congestion Control: Circuit Breakers for Unicast RTP Sessions," RFC 8083, Mar. 2017. [Online]. Available: https://rfc-editor.org/rfc/rfc8083.txt

[33] V. Singh, J. Ott, and S. Holmer, "Evaluating Congestion Control for Interactive Real-time Media," Internet Engineering Task Force, Internet-Draft draft-ietf-rmcat-eval-criteria-09, Jul. 2019, work in Progress. [Online]. Available: https://datatracker.ietf.org/doc/html/draft-ietf-rmcat-eval-criteria-09

[34] W. Lei, W. Zhang, and S. Liu, "Multipath Real-Time Transport Protocol Based on Application-Level Relay (MPRTP-AR)," Internet Engineering Task Force, Internet-Draft draft-leiwm-avtcore-mprtp-ar-09, Feb. 2018, work in Progress. [Online]. Available: https://datatracker.ietf.org/doc/html/draft-leiwm-avtcore-mprtp-ar-09

[35] "Transmission Control Protocol," RFC 793, Sep. 1981. [Online]. Available: https://rfc-editor.org/rfc/rfc793.txt

[36] C. J. Kale and T. J. Socolofsky, "TCP/IP tutorial," RFC 1180, Jan. 1991. [Online]. Available: https://rfc-editor.org/rfc/rfc1180.txt

[37] H. F. Nielsen, J. Mogul, L. M. Masinter, R. T. Fielding, J. Gettys, P. J. Leach, and T. Berners-Lee, "Hypertext Transfer Protocol – HTTP/1.1," RFC 2616, Jun. 1999. [Online]. Available: https://rfc-editor.org/rfc/rfc2616.txt

[38] R. Hamilton, J. Iyengar, I. Swett, and A. Wilk, "QUIC: A UDP-Based Secure and Reliable Transport for HTTP/2," Internet Engineering Task Force, Internet-Draft draft-tsvwg-quic-protocol-02, Jan. 2016, work in Progress. [Online]. Available: https://datatracker.ietf.org/doc/html/draft-tsvwg-quic-protocol-02

[39] A. Langley, A. Riddoch, A. Wilk, A. Vicente, C. Krasic, D. Zhang, F. Yang, F. Kouranov, I. Swett, J. Iyengar, J. Bailey, J. Dorfman, J. Roskind, J. Kulik, P. Westin, R. Tenneti, R. Shade, R. Hamilton, V. Vasiliev, W.-T. Chang, and Z. Shi, "The QUIC Transport Protocol: Design and Internet-Scale Deployment," in *Proceedings of the Conference of the ACM Special Interest Group on Data Communication*, ser. SIGCOMM '17. New York, NY, USA: ACM, 2017, pp. 183–196. [Online]. Available: http://doi.acm.org/10.1145/3098822.3098842

[40] N. Cardwell, Y. Cheng, C. S. Gunn, S. H. Yeganeh, and V. Jacobson, "BBR: Congestion-Based Congestion Control," *Queue*, vol. 14, no. 5, pp. 50:20–50:53, Oct. 2016. [Online]. Available: http://doi.acm.org/10.1145/3012426.3022184

[41] I. Sodagar, "The MPEG-DASH Standard for Multimedia Streaming Over the Internet," *IEEE MultiMedia*, Apr. 2011.

[42] G. Paschos, E. Bastug, I. Land, G. Caire, and M. Debbah, "Wireless caching: technical misconceptions and business barriers," *IEEE Communications Magazine*, vol. 54, no. 8, pp. 16–22, August 2016.

[43] Open IPTV Forum (OIPF) Release 2 Specification Volume 2AHTTP Adaptive Streaming. [Online]. Available: https://www.oipf.tv/docs/OIPF-T1-R2-Specification-Volume-2a-HTTP-Adaptive-Streaming-v2_3-2014-01-24.pdf

[44] Digital Video Broadcasting (DVB); MPEGDASH Profile for Transport of ISO BMFF Based DVB Services Over IP Based Networks. [Online]. Available: https://www.dvb.org/resources/public/standards/a168_dvb-dash.pdf

[45] DASH Industry Forum, "Guidelines for Implementation: DASH-AVC/264 Test cases and Vectors," *report*, Jan. 2014.

[46] T. Hoßfeld, P. E. Heegaard, L. Skorin-Kapov, and M. Varela, "Fundamental Relationships for Deriving QoE in Systems," in *2019 Eleventh International Conference on Quality of Multimedia Experience (QoMEX)*, June 2019, pp. 1–6.

[47] Z. Ye, R. Elazouzi, and T. Jiménez, "Quality of Experience in HTTP Adaptive Video Streaming Systems," in *Ubiquitous Networking*. Springer International Publishing, 2017.

[48] M. Seufert, S. Egger, M. Slanina, T. Zinner, T. Hoßfeld, and P. Tran-Gia, "A Survey on Quality of Experience of HTTP Adaptive Streaming," *IEEE Communications Surveys Tutorials*, Feb. 2015.

[49] S. Poojary, R. El-Azouzi, E. Altman, A. Sunny, I. Triki, M. Haddad, T. Jimenez, S. Valentin, and D. Tsilimantos, "Analysis of QoE for adaptive video streaming over wireless networks," in *2018 16th International Symposium on Modeling and Optimization in Mobile, Ad Hoc, and Wireless Networks (WiOpt)*, May 2018, pp. 1–8.

[50] T. Hoßfeld, S. Egger, R. Schatz, M. Fiedler, K. Masuch, and C. Lorentzen, "Initial delay vs. interruptions: Between the devil and the deep blue sea," in *2012 Fourth International Workshop on Quality of Multimedia Experience*, July 2012.

[51] T. Hoßfeld, M. Seufert, M. Hirth, T. Zinner, P. Tran-Gia, and R. Schatz, "Quantification of YouTube QoE via Crowdsourcing," in *2011 IEEE International Symposium on Multimedia*, Dec 2011, pp. 494–499.

[52] A. Aldahdooh, E. Masala, G. V. Wallendael, and M. Barkowsky, "Reproducible research framework for objective video quality measures using a large-scale database approach," *SoftwareX*, vol. 8, pp. 64 – 68, 2018, digital Signal Processing & SoftwareX - Joint Special Issue on Reproducible Research in Signal Processing. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S2352711017300468

[53] A. Aldahdooh, E. Masala, O. Janssens, G. Van Wallendael, M. Barkowsky, and P. Le Callet, "Improved Performance Measures for Video Quality Assessment Algorithms Using Training and Validation Sets," *IEEE Transactions on Multimedia*, vol. 21, no. 8, pp. 2026–2041, Aug 2019.

[54] J. Y. Lin, Chi-Hao Wu, I. Katsavounidis, Zhi Li, A. Aaron, and C. . J. Kuo, "EVQA: An ensemble-learning-based video quality assessment index," in *2015 IEEE International Conference on Multimedia Expo Workshops (ICMEW)*, June 2015, pp. 1–6.

[55] Toward A Practical Perceptual Video Quality Metric. [Online]. Available: https://medium.com/netflix-techblog/toward-a-practical-perceptual-video-quality-metric-653f208b9652

[56] D. P. Bertsekas, *Convex Analysis and Optimization.* Athena Scientific, 2003.

[57] T. Karagkioules, G. S. Paschos, N. Liakopoulos, A. Fiandrotti, D. Tsilimantos, and M. Cagnazzo, "Online Learning for Robust Adaptive Video Streaming in Mobile Networks," *CoRR*, vol. abs/1905.11705, 2019. [Online]. Available: http://arxiv.org/abs/1905.11705

[58] J. Kua, G. Armitage, and P. Branch, "A Survey of Rate Adaptation Techniques for Dynamic Adaptive Streaming Over HTTP," *IEEE Communications Surveys Tutorials*, vol. 19, no. 3, pp. 1842–1866, thirdquarter 2017.

[59] Y. Sani, A. Mauthe, and C. Edwards, "Adaptive Bitrate Selection: A Survey," *IEEE Communications Surveys Tutorials*, vol. 19, no. 4, pp. 2985–3014, Fourthquarter 2017.

[60] A. Bentaleb, B. Taani, A. C. Begen, C. Timmerer, and R. Zimmermann, "A Survey on Bitrate Adaptation Schemes for Streaming Media Over HTTP," *IEEE Communications Surveys Tutorials*, vol. 21, no. 1, pp. 562–585, Firstquarter 2019.

[61] Z. Li, X. Zhu, J. Gahm, R. Pan, H. Hu, A. C. Begen, and D. Oran, "Probe and Adapt: Rate Adaptation for HTTP Video Streaming At Scale," *IEEE JSAC*, Apr. 2014.

[62] C. Liu, I. Bouazizi, and M. Gabbouj, "Rate Adaptation for Adaptive HTTP Streaming," in *Proceedings of the Second Annual ACM Conference on Multimedia Systems*, ser. MMSys '11. New York, NY, USA: ACM, 2011, pp. 169–174. [Online]. Available: http://doi.acm.org/10.1145/1943552.1943575

[63] C. Liu, I. Bouazizi, M. M. Hannuksela, and M. Gabbouj, "Rate adaptation for dynamic adaptive streaming over HTTP in content distribution network," *Signal Processing: Image Communication*, vol. 27, no. 4, pp. 288 – 311, 2012.

[64] J. Jiang, V. Sekar, and H. Zhang, "Improving Fairness, Efficiency, and Stability in HTTP-Based Adaptive Video Streaming With Festive," *IEEE/ACM Transactions on Networking*, vol. 22, no. 1, pp. 326–340, Feb 2014.

[65] K. Miller, A.-K. Al-Tamimi, and A. Wolisz, "QoE-Based Low-Delay Live Streaming Using Throughput Predictions," *ACM Trans. Multimedia Comput. Commun. Appl.*, vol. 13, no. 1, pp. 4:1–4:24, Oct. 2016. [Online]. Available: http://doi.acm.org/10.1145/2990505

[66] M. Xiao, V. Swaminathan, S. Wei, and S. Chen, "DASH2M: Exploring HTTP/2 for Internet Streaming to Mobile Devices," in *Proceedings of the 24th ACM International Conference on Multimedia*, ser. MM '16. New York, NY, USA: ACM, 2016, pp. 22–31. [Online]. Available: http://doi.acm.org/10.1145/2964284.2964313

[67] M. B. Yahia, Y. L. Louedec, G. Simon, L. Nuaymi, and X. Corbillon, "HTTP/2-based Frame Discarding for Low-Latency Adaptive Video Streaming," *ACM Trans. Multimedia Comput. Commun. Appl.*, vol. 15, no. 1, pp. 18:1–18:23, Feb. 2019. [Online]. Available: http://doi.acm.org/10.1145/3280854

[68] T.-Y. Huang, R. Johari, N. McKeown, M. Trunnell, and M. Watson, "A Buffer-based Approach to Rate Adaptation: Evidence from a Large Video Streaming Service," in *Proc. ACM SIGCOMM*, Aug. 2014.

[69] K. Spiteri, R. Urgaonkar, and R. K. Sitaraman, "BOLA: Near-optimal bitrate adaptation for online videos," in *IEEE Int. Conf. on INFOCOM.*, April 2016.

[70] M. J. Neely, *Stochastic Network Optimization with Application to Communication and Queueing Systems.* Morgan and Claypool Publishers, 2010.

[71] K. Spiteri, R. Sitaraman, and D. Sparacio, "From Theory to Practice: Improving Bitrate Adaptation in the DASH Reference Player," *ACM Trans. Multimedia Comput. Commun. Appl.*, vol. 15, no. 2s, pp. 67:1–67:29, Jul. 2019. [Online]. Available: http://doi.acm.org/10.1145/3336497

[72] P. K. Yadav, A. Shafiei, and W. T. Ooi, "QUETRA: A Queuing Theory Approach to DASH Rate Adaptation," in *Proceedings of the 25th ACM International Conference on Multimedia*, ser. MM '17. New York, NY, USA: ACM, 2017, pp. 1130–1138. [Online]. Available: http://doi.acm.org/10.1145/3123266.3123390

[73] L. Kleinrock, *Theory, Volume 1, Queueing Systems.* New York, NY, USA: Wiley-Interscience, 1975.

[74] V. Burger, T. Zinner, L. Dinh-Xuan, F. Wamser, and P. Tran-Gia, "A Generic Approach to Video Buffer Modeling Using Discrete-Time Analysis," *ACM Trans. Multimedia Comput. Commun. Appl.*,

vol. 14, no. 2s, pp. 33:1–33:23, Apr. 2018. [Online]. Available: http://doi.acm.org/10.1145/3183511

[75] K. Miller, E. Quacchio, G. Gennari, and A. Wolisz, "Adaptation algorithm for adaptive streaming over HTTP," in *2012 19th International Packet Video Workshop (PV)*, May 2012, pp. 173–178.

[76] P. Juluri, V. Tamarapalli, and D. Medhi, "SARA: Segment aware rate adaptation algorithm for dynamic adaptive streaming over HTTP," in *2015 IEEE International Conference on Communication Workshop (ICCW)*, June 2015, pp. 1765–1770.

[77] S. Mekki, T. Karagkioules, and S. Valentin, "HTTP adaptive streaming with indoors-outdoors detection in mobile networks," in *IEEE INFOCOM Workshops*, May 2017.

[78] L. Toni, R. Aparicio-Pardo, K. Pires, G. Simon, A. Blanc, and P. Frossard, "Optimal Selection of Adaptive Streaming Representations," *ACM Trans. Multimedia Comput. Commun. Appl.*, vol. 11, no. 2s, pp. 43:1–43:26, Feb. 2015. [Online]. Available: http://doi.acm.org/10.1145/2700294

[79] A. Beben, P. Wiśniewski, J. M. Batalla, and P. Krawiec, "ABMA+: Lightweight and Efficient Algorithm for HTTP Adaptive Streaming," in *Proc. Int. ACM Conference on Multimedia Systems (MMSys)*, May 2016, pp. 2:1–2:11.

[80] D. P. Bertsekas, *Dynamic Programming and Optimal Control*. Athena Scientific, 1976.

[81] A. Bokani, M. Hassan, S. Kanhere, and X. Zhu, "Optimizing HTTP-Based Adaptive Streaming in Vehicular Environment Using Markov Decision Process," *IEEE Transactions on Multimedia*, vol. 17, no. 12, pp. 2297–2309, Dec 2015.

[82] W. Bao and S. Valentin, "Bitrate Adaptation for Mobile Video Streaming Based on Buffer and Channel State," in *Proc. IEEE Int. Conf. on Commun. (ICC)*, Jun. 2015.

[83] M. Xing, S. Xiang, and L. Cai, "A Real-Time Adaptive Algorithm for Video Streaming over Multiple Wireless Access Networks," *IEEE Journal on Selected Areas in Communications*, vol. 32, no. 4, pp. 795–805, April 2014.

[84] A. Bentaleb, A. C. Begen, S. Harous, and R. Zimmermann, "Want to Play DASH?: A Game Theoretic Approach for Adaptive Streaming over HTTP," in *Proceedings of the 9th ACM Multimedia Systems*

Conference, ser. MMSys '18. New York, NY, USA: ACM, 2018, pp. 13–26. [Online]. Available: http://doi.acm.org/10.1145/3204949.3204961

[85] N. Changuel, B. Sayadi, and M. Kieffer, "Online learning for QoE-based video streaming to mobile receivers," in *2012 IEEE Globecom Workshops*, Dec 2012, pp. 1319–1324.

[86] L. Georgiadis, M. Neely, and L. Tassiulas, *Resource Allocation and Cross Layer Control in Wireless Networks (Foundations and Trends in Networking, V. 1, No. 1)*. Hanover, MA, USA: Now Publishers Inc., 2006.

[87] R. Sutton and A. Barto, *Reinforcement Learning: An Introduction*. MIT Press, Cambridge, MA, 1998.

[88] F. Chiariotti, S. D'Aronco, L. Toni, and P. Frossard, "Online Learning Adaptation Strategy for DASH Clients," in *Proceedings of the 7th International Conference on Multimedia Systems*, ser. MMSys '16. New York, NY, USA: ACM, 2016, pp. 8:1–8:12. [Online]. Available: http://doi.acm.org/10.1145/2910017.2910603

[89] R. K. P. Mok, X. Luo, E. W. W. Chan, and R. K. C. Chang, "QDASH: A QoE-aware DASH System," in *Proceedings of the 3rd Multimedia Systems Conference*, ser. MMSys '12. New York, NY, USA: ACM, 2012, pp. 11–22. [Online]. Available: http://doi.acm.org/10.1145/2155555.2155558

[90] N. Bouten, R. d. O. Schmidt, J. Famaey, S. Latré, A. Pras, and F. De Turck, "QoE-driven In-network Optimization for Adaptive Video Streaming Based on Packet Sampling Measurements," *Comput. Netw.*, vol. 81, no. C, pp. 96–115, Apr. 2015. [Online]. Available: http://dx.doi.org/10.1016/j.comnet.2015.02.007

[91] S. D'Aronco, L. Toni, and P. Frossard, "Price-Based Controller for Utility-Aware HTTP Adaptive Streaming," *IEEE MultiMedia*, vol. 24, no. 2, pp. 20–29, Apr 2017.

[92] E. Gourdin, P. Maill, G. Simon, and B. Tuffin, "The Economics of CDNs and Their Impact on Service Fairness," *IEEE Transactions on Network and Service Management*, vol. 14, no. 1, pp. 22–33, March 2017.

[93] K. He, P. Maillé, and G. Simon, "Watermarked Video Delivery: Traffic Reduction and CDN Management," in *Proceedings of the 9th ACM Multimedia Systems Conference*, ser. MMSys '18. New York, NY, USA: ACM, 2018, pp. 76–88. [Online]. Available: http://doi.acm.org/10.1145/3204949.3204964

[94] A. V. Vasilakos, Z. Li, G. Simon, and W. You, "Information centric network: Research challenges and opportunities," *Journal of Network and Computer Applications*, vol. 52, pp. 1 – 10, 2015. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S1084804515000272

[95] B. Baccala, "Data-oriented networking ," Internet Engineering Task Force, Internet-Draft draft-baccala-data-networking-00, Aug. 2002, work in Progress. [Online]. Available: https://datatracker.ietf.org/doc/html/draft-baccala-data-networking-00

[96] J. Samain, G. Carofiglio, L. Muscariello, M. Papalini, M. Sardara, M. Tortelli, and D. Rossi, "Dynamic Adaptive Video Streaming: Towards a Systematic Comparison of ICN and TCP/IP," *IEEE Transactions on Multimedia*, vol. 19, no. 10, pp. 2166–2181, Oct 2017.

[97] J. Samain, G. Carofiglio, M. Tortelli, and D. Rossi, "A Simple Yet Effective Network-assisted Signal for Enhanced DASH Quality of Experience," in *Proceedings of the 28th ACM SIGMM Workshop on Network and Operating Systems Support for Digital Audio and Video*, ser. NOSSDAV '18. New York, NY, USA: ACM, 2018, pp. 55–60. [Online]. Available: http://doi.acm.org/10.1145/3210445.3210447

[98] E. Thomas, M. O. van Deventer, T. Stockhammer, A. C. Begen, M. Champel, and O. Oyman, "Application of SAND Technology in DASH-Enabled Content Delivery Networks and Server Environments," *SMPTE Motion Imaging Journal*, vol. 127, no. 1, pp. 48–54, Jan 2018.

[99] R. Jmal, G. Simon, and L. Chaari, "Network-assisted strategy for dash over CCN," in *2017 IEEE International Conference on Multimedia and Expo (ICME)*, July 2017, pp. 13–18.

[100] V. Jacobson, D. K. Smetters, J. D. Thornton, M. F. Plass, N. H. Briggs, and R. L. Braynard, "Networking Named Content," in *Proceedings of the 5th International Conference on Emerging Networking Experiments and Technologies*, ser. CoNEXT '09. New York, NY, USA: ACM, 2009, pp. 1–12. [Online]. Available: http://doi.acm.org/10.1145/1658939.1658941

[101] B. Han, F. Qian, L. Ji, and V. Gopalakrishnan, "MP-DASH: Adaptive Video Streaming Over Preference-Aware Multipath," in *Proceedings of the 12th International on Conference on Emerging Networking EXperiments and Technologies*, ser. CoNEXT '16. New York, NY, USA: ACM, 2016, pp. 129–143. [Online]. Available: http://doi.acm.org/10.1145/2999572.2999606

[102] C. James, E. Halepovic, M. Wang, R. Jana, and N. K. Shankaranarayanan, "Is Multipath TCP (MPTCP) Beneficial for Video Streaming over DASH?" in *2016 IEEE 24th International Symposium on Modeling, Analysis and Simulation of Computer and Telecommunication Systems (MASCOTS)*, Sep. 2016, pp. 331–336.

[103] V. Poliakov, L. Sassatelli, and D. Saucez, "Adaptive Video Streaming, Multipath and Caching: Can Less Be More?" in *2018 IEEE International Conference on Communications (ICC)*, May 2018, pp. 1–6.

[104] G. Cofano, L. De Cicco, T. Zinner, A. Nguyen-Ngoc, P. Tran-Gia, and S. Mascolo, "Design and Experimental Evaluation of Network-assisted Strategies for HTTP Adaptive Streaming," in *Proceedings of the 7th International Conference on Multimedia Systems*, ser. MMSys '16. New York, NY, USA: ACM, 2016, pp. 3:1–3:12. [Online]. Available: http://doi.acm.org/10.1145/2910017.2910597

[105] S. Akhshabi, L. Anantakrishnan, C. Dovrolis, and A. C. Begen, "Server-based Traffic Shaping for Stabilizing Oscillating Adaptive Streaming Players," in *Proceeding of the 23rd ACM Workshop on Network and Operating Systems Support for Digital Audio and Video*, ser. NOSSDAV '13. New York, NY, USA: ACM, 2013, pp. 19–24. [Online]. Available: http://doi.acm.org/10.1145/2460782.2460786

[106] R. Houdaille and S. Gouache, "Shaping HTTP Adaptive Streams for a Better User Experience," in *Proceedings of the 3rd Multimedia Systems Conference*, ser. MMSys '12. New York, NY, USA: ACM, 2012, pp. 1–9. [Online]. Available: http://doi.acm.org/10.1145/2155555.2155557

[107] A. Detti, B. Ricci, and N. Blefari-Melazzi, "Tracker-assisted rate adaptation for MPEG DASH live streaming," in *IEEE INFOCOM 2016 - The 35th Annual IEEE International Conference on Computer Communications*, April 2016, pp. 1–9.

[108] L. De Cicco, S. Mascolo, and V. Palmisano, "Feedback Control for Adaptive Live Video Streaming," in *Proceedings of the Second Annual ACM Conference on Multimedia Systems*, ser. MMSys '11. New York, NY, USA: ACM, 2011, pp. 145–156. [Online]. Available: http://doi.acm.org/10.1145/1943552.1943573

[109] A. Sunny, R. El-Azouzi, A. Arfaoui, E. Altman, S. Poojary, D. Tsilimantos, and S. Valentin, "Enforcing Bitrate-Stability for Adaptive Streaming Traffic in Cellular Networks," *IEEE Transactions on Network and Service Management*, pp. 1–1, 2019.

[110] C. Timmerer, M. Maiero, and B. Rainer, "Which Adaptation Logic? An Objective and Subjective Performance Evaluation of HTTP-based Adaptive Media Streaming Systems," *CoRR*, vol. abs/1606.00341, 2016. [Online]. Available: http://arxiv.org/abs/1606.00341

[111] C. Müller, S. Lederer, and C. Timmerer, "An Evaluation of Dynamic Adaptive Streaming over HTTP in Vehicular Environments," in *Proceedings of the 4th Workshop on Mobile Video*, ser. MoVid '12. New York, NY, USA: ACM, 2012, pp. 37–42. [Online]. Available: http://doi.acm.org/10.1145/2151677.2151686

[112] S. Akhshabi, A. C. Begen, and C. Dovrolis, "An Experimental Evaluation of Rate-adaptation Algorithms in Adaptive Streaming over HTTP," in *Proceedings of the Second Annual ACM Conference on Multimedia Systems*, ser. MMSys '11. New York, NY, USA: ACM, 2011, pp. 157–168. [Online]. Available: http://doi.acm.org/10.1145/1943552.1943574

[113] T. C. Thang, H. T. Le, A. T. Pham, and Y. M. Ro, "An Evaluation of Bitrate Adaptation Methods for HTTP Live Streaming," *IEEE Journal on Selected Areas in Communications*, vol. 32, no. 4, pp. 693–705, April 2014.

[114] A. Zabrovskiy, E. Petrov, E. Kuzmin, and C. Timmerer, "Evaluation of the Performance of Adaptive HTTP Streaming Systems," *CoRR*, vol. abs/1710.02459, 2017. [Online]. Available: http://arxiv.org/abs/1710.02459

[115] H. Riiser, H. S. Bergsaker, P. Vigmostad, P. Halvorsen, and C. 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*, ser. MoVid '12. New York, NY, USA: ACM, 2012, pp. 25–30. [Online]. Available: http://doi.acm.org/10.1145/2151677.2151684

[116] A. Zabrovskiy, E. Kuzmin, E. Petrov, C. Timmerer, and C. Mueller, "AdViSE: Adaptive Video Streaming Evaluation Framework for the Automated Testing of Media Players," in *Proceedings of the 8th ACM on Multimedia Systems Conference*, ser. MMSys'17. New York, NY, USA: ACM, 2017, pp. 217–220. [Online]. Available: http://doi.acm.org/10.1145/3083187.3083221

[117] D. Raca, Y. Sani, C. J. Sreenan, and J. J. Quinlan, "DASHbed: A Testbed Framework for Large Scale Empirical Evaluation of Real-time DASH in Wireless Scenarios," in *Proceedings of the 10th ACM Multimedia Systems Conference*, ser. MMSys '19. New

York, NY, USA: ACM, 2019, pp. 285–290. [Online]. Available: http://doi.acm.org/10.1145/3304109.3325813

[118] J. Le Feuvre, C. Concolato, and J.-C. Moissinac, "GPAC: Open Source Multimedia Framework," in *Proceedings of the 15th ACM International Conference on Multimedia*, ser. MM '07. New York, NY, USA: ACM, 2007, pp. 1009–1012. [Online]. Available: http://doi.acm.org/10.1145/1291233.1291452

[119] C. G. P. H. H. Riiser, P. Vigmostad, "Commute Path Bandwidth Traces from 3G Networks: Analysis and Applications," *Proc. of MMSys*, vol. 5, no. 1, pp. 114–118, Mar. 2013.

[120] ITEC - Dynamic Adaptive Streaming over HTTP. URL accessed on December 22, 2016. [Online]. Available: http://www-itec.uni-klu.ac.at/ftp/datasets/DASHDataset2014/BigBuckBunny/

[121] E. Liotou, T. Hoßfeld, C. Moldovan, F. Metzger, D. Tsolkas, and N. Passas, "Enriching HTTP adaptive streaming with context awareness: A tunnel case study," in *Proc. IEEE Int. Conf. on Commun. (ICC)*, May 2016, pp. 1–6.

[122] I. Triki, M. Haddad, R. El-Azouzi, A. Feki, and M. Gachaoui, "Context-aware mobility resource allocation for QoE-driven streaming services," in *2016 IEEE Wireless Communications and Networking Conference*, April 2016, pp. 1–6.

[123] J. Yao, S. S. Kanhere, and M. Hassan, "Improving QoS in High-Speed Mobility Using Bandwidth Maps," *IEEE Transactions on Mobile Computing*, vol. 11, no. 4, pp. 603–617, April 2012.

[124] X. Xie, X. Zhang, S. Kumar, and L. E. Li, "piStream: Physical Layer Informed Adaptive Video Streaming over LTE," in *Proceedings of the 21st Annual International Conference on Mobile Computing and Networking*, ser. MobiCom '15. New York, NY, USA: ACM, 2015, pp. 413–425. [Online]. Available: http://doi.acm.org/10.1145/2789168.2790118

[125] T. M. Cover and J. A. Thomas, *Elements of Information Theory (Wiley Series in Telecommunications and Signal Processing)*. New York, NY, USA: Wiley-Interscience, 2006.

[126] S. Sesia, I. Toufik, and M. Baker, *LTE, The UMTS Long Term Evolution: From Theory to Practice*. Wiley Publishing, 2009.

[127] Google Inc., "Signal Strength," 2016, URL verified on March 12, 2017. [Online]. Available: https://developer.android.com/reference/android/telephony/SignalStrength.html

[128] E. T. S. I. (ETSI), "Digital cellular telecommunications system (Phase 2+); Radio transmission and reception," GSMTS, Tech. Rep. GSM 05.05 Version 5.0.0, March 1996.

[129] ——, "Universal Mobile Telecommunications System (UMTS); User Equipment (UE) radio transmission and reception (FDD)," 3GPP, TS 25.101 version 11.3.0 Release 11, Nov. 2012.

[130] ——, "LTE; Evolved Universal Terrestrial Radio Access (E-UTRA); User Equipment (UE) conformance specification; Radio transmission and reception; Part 1: Conformance testing," 3GPP, TS 36.521-1 version 13.0.1 Release 13, May 2016.

[131] ——, "Digital cellular telecommunications system (Phase 2+); Physical layer on the radio path; General description," GSMTS, Tech. Rep. GSM 05.01 Version 5.0.0, May 1996.

[132] Google Inc., "Android 6.0 Marshmallow," 2016, URL verified on March 12, 2017. [Online]. Available: https://www.android.com/intl/en_uk/versions/marshmallow-6-0/

[133] E. T. S. I. (ETSI), "LTE; Evolved Universal Terrestrial Radio Access (E-UTRA); Physical layer - Measurements," 3GPP, TS 36.214 version 9.1.0 Release 9, April 2010.

[134] A. Goldsmith, *Wireless Communications*. New York, NY, USA: Cambridge University Press, 2005.

[135] Google Inc., "Location Strategies," 2016, URL verified on March 12, 2017. [Online]. Available: https://developer.android.com/guide/topics/location/strategies.html

[136] B. Foundation. (2008, May) Big buck bunny. Video. [Online]. Available: https://www.youtube.com/watch?v=YE7VzlLtp-4

[137] M. Gadaleta, F. Chiariotti, M. Rossi, and A. Zanella, "D-DASH: A Deep Q-Learning Framework for DASH Video Streaming," *IEEE Transactions on Cognitive Communications and Networking*, vol. 3, no. 4, pp. 703–718, Dec 2017.

[138] X. Yin, A. Jindal, V. Sekar, and B. Sinopoli, "A Control-Theoretic Approach for Dynamic Adaptive Video Streaming over HTTP," in *Proc. ACM Conf. on SIGCOMM*, August 2015, pp. 325–338. [Online]. Available: http://doi.acm.org/10.1145/2785956.2787486

[139] M. Zinkevich, "Online Convex Programming and Generalized Infinitesimal Gradient Ascent," in *Proc. of Inter. Conf. Machine Learning*, ser. ICML'03, 2003. [Online]. Available: http://dl.acm.org/citation.cfm?id=3041838.3041955

[140] E. V. Belmega, P. Mertikopoulos, R. Negrel, and L. Sanguinetti, "Online convex optimization and no-regret learning: Algorithms, guarantees and applications," *corr*, 2018. [Online]. Available: http://arxiv.org/abs/1804.04529

[141] N. Liakopoulos, G. Paschos, and T. Spyropoulos, "No Regret in Cloud Resources Reservation with Violation Guarantees," in *Proc. IEEE INFOCOM*, May 2019.

[142] T. Chen, Q. Ling, and G. B. Giannakis, "An Online Convex Optimization Approach to Proactive Network Resource Allocation," *IEEE Transactions on Signal Processing*, Dec. 2017.

[143] S. Kim and C. Kim, "XMAS: An Efficient Mobile Adaptive Streaming Scheme Based on Traffic Shaping," *IEEE Transactions on Multimedia*, vol. 21, no. 2, Feb 2019.

[144] C. Zhou, C. Lin, and Z. Guo, "mDASH: A Markov Decision-Based Rate Adaptation Approach for Dynamic HTTP Streaming," *IEEE Transactions on Multimedia*, April 2016.

[145] M. Claeys, S. Latr, J. Famaey, T. Wu, W. Van Leekwijck, and F. De Turck, "Design of a Q-learning-based client quality selection algorithm for HTTP adaptive video streaming," in *In Proc. of Adaptive and Learning Agents Workshop, part of AAMAS*, May 2013.

[146] H. Mao, R. Netravali, and M. Alizadeh, "Neural Adaptive Video Streaming with Pensieve," in *Proc. of ACM Conf. Special Interest Group on Data Communication*, ser. SIGCOMM '17, 2017.

[147] T.-Y. Huang, C. Ekanadham, A. J. Berglund, and Z. Li, "Hindsight: Evaluate Video Bitrate Adaptation at Scale," in *Proceedings of the 10th ACM Multimedia Systems Conference*, ser. MMSys '19. New York, NY, USA: ACM, 2019, pp. 86–97. [Online]. Available: http://doi.acm.org/10.1145/3304109.3306219

[148] D. Tsilimantos, T. Karagkioules, and S. Valentin, "Classifying Flows and Buffer State for Youtube's HTTP Adaptive Streaming Service in Mobile Networks," in *Proceedings of the 9th ACM Multimedia Systems Conference*, ser. MMSys '18. New York, NY, USA: ACM, 2018, pp. 138–149. [Online]. Available: http://doi.acm.org/10.1145/3204949.3204955

[149] N. Liakopoulos, A. Destounis, G. Paschos, T. Spyropoulos, and P. Mertikopoulos, "Cautious Regret Minimization: Online Optimization with Long-Term Budget Constraints," in *Proc. of ICML*, Jun. 2019.

[150] M. J. Neely and H. Yu, "Online Convex Optimization with Time-Varying Constraints," *arXiv e-prints*, Feb. 2017.

[151] S. Shalev-Shwartz, "Online Learning and Online Convex Optimization," *Found. Trends Mach. Learn.*, Feb. 2012.

[152] D. Raca, J. J. Quinlan, A. H. Zahran, and C. J. Sreenan, "Beyond Throughput: A 4G LTE Dataset with Channel and Context Metrics," in *Proc. of ACM*, ser. MMSys '18, Jun. 2018.

[153] A. Zabrovskiy, C. Feldmann, and C. Timmerer, "Multi-codec DASH Dataset," in *Proc. of ACM*, ser. MMSys '18, Jun. 2018.

[154] V. Krishnamoorthi, N. Carlsson, E. Halepovic, and E. Petajan, "BUFFEST: Predicting Buffer Conditions and Real-time Requirements of HTTP(S) Adaptive Streaming Clients," in *Proc. ACM MMSys*, Jan. 2017, pp. 76–87.

[155] D. Tsilimantos, A. Nogales-Gómez, and S. Valentin, "Anticipatory Radio Resource Management for Mobile Video Streaming with Linear Programming," in *IEEE ICC*, May 2016, pp. 1–6.

[156] F. Wamser, P. Casas, M. Seufert, C. Moldovan, P. Tran-Gia, and T. Hoßfeld, "Modeling the YouTube Stack: from Packets to Quality of Experience," *Computer Networks*, vol. 109, no. Part 2, pp. 211–224, Apr. 2016.

[157] A. Rao, A. Legout, Y.-s. Lim, D. Towsley, C. Barakat, and W. Dabbous, "Network Characteristics of Video Streaming Traffic," in *Proc. ACM CoNEXT*, Dec. 2011, pp. 1–12.

[158] S. Sengupta et al. (2015) CRAWDAD dataset (v. 2015-11-26). Dataset. [Online]. Available: https://crawdad.org/iitkgp/apptraffic/20151126/

[159] A. Mansy, M. Ammar, J. Chandrashekar, and A. Sheth, "Characterizing Client Behavior of Commercial Mobile Video Streaming Services," in *Proceedings of Workshop on Mobile Video Delivery*, ser. MoViD'14. New York, NY, USA: ACM, 2013, pp. 8:1–8:6. [Online]. Available: http://doi.acm.org/10.1145/2594449.2579469

[160] P. Ameigeiras, J. J. Ramos-Munoz, J. Navarro-Ortiz, and J. Lopez-Soler, "Analysis and modelling of YouTube traffic," *Trans. Emerging Telecommunications Technologies*, vol. 23, no. 4, pp. 360–377, 2012.

[161] J. J. Ramos-Munoz, J. Prados-Garzon, P. Ameigeiras, J. Navarro-Ortiz, and J. M. Lopez-Soler, "Characteristics of Mobile YouTube Traffic," *IEEE Wireless Communications*, vol. 21, no. 1, pp. 18–25, Feb. 2014.

[162] Y. Liu, S. Dey, F. Ulupinar, M. Luby, and Y. Mao, "Deriving and Validating User Experience Model for DASH Video Streaming," *IEEE Trans. Broadcast.*, vol. 61, no. 4, pp. 651–665, Dec. 2015.

[163] C. Sieber, A. Blenk, M. Hinteregger, and W. Kellerer, "The cost of aggressive HTTP adaptive streaming: Quantifying YouTube's redundant traffic," in *Proc. IFIP/IEEE IM'15*, May 2015, pp. 1261–1267.

[164] T. T. T. Nguyen and G. Armitage, "A survey of techniques for Internet traffic classification using machine learning," *IEEE Communications Surveys Tutorials*, vol. 10, no. 4, pp. 56–76, Fourth 2008.

[165] G. Dimopoulos, P. Barlet-Ros, and J. Sanjuàs-Cuxart, "Analysis of YouTube user experience from passive measurements," in *Proc. IEEE CNSM*, Oct. 2013, pp. 260–267.

[166] I. Orsolic, D. Pevec, M. Suznjevic, and L. Skorin-Kapov, "A machine learning approach to classifying YouTube QoE based on encrypted network traffic," *Springer Multimedia Tools and Applications*, vol. 76, no. 21, pp. 22 267–22 301, Nov. 2017.

[167] G. Dimopoulos, I. Leontiadis, P. Barlet-Ros, and K. Papagiannaki, "Measuring Video QoE from Encrypted Traffic," in *Proc. ACM IMC*, Nov. 2016, pp. 513–526.

[168] M. Katsarakis, R. C. Teixeira, M. Papadopouli, and V. Christophides, "Towards a Causal Analysis of Video QoE from Network and Application QoS," in *Proc. ACM Internet-QoE*, Aug. 2016, pp. 31–36.

[169] S. Galetto, P. Bottaro, C. Carrara, F. Secco, A. Guidolin, E. Targa, C. Narduzzi, and G. Giorgi, "Detection of video/audio streaming packet flows for non-intrusive QoS/QoE monitoring," in *Proc. IEEE MN*, Sep. 2017, pp. 1–6.

[170] T. Mangla, E. Halepovic, M. Ammar, and E. Zegura, "Using Session Modeling to Estimate HTTP-Based Video QoE Metrics From Encrypted Network Traffic," *IEEE Transactions on Network and Service Management*, vol. 16, no. 3, pp. 1086–1099, Sep 2019.

[171] ——, "eMIMIC: Estimating HTTP-Based Video QoE Metrics from Encrypted Network Traffic," in *2018 Network Traffic Measurement and Analysis Conference (TMA)*, June 2018, pp. 1–8.

[172] D. Tsilimantos, T. Karagkioules, A. Nogales-Gómez, and S. Valentin, "Traffic profiling for mobile video streaming," in *2017 IEEE International Conference on Communications (ICC)*, May 2017.

[173] M. Seufert, B. Zeidler, F. Wamser, T. Karagkioules, D. Tsilimantos, F. Loh, P. Tran-Gia, and S. Valentin, "A Wrapper for Automatic Measurements with YouTube's Native Android App," in *Proc. IEEE TMA*, Jun. 2018.

[174] T. Karagkioules, D. Tsilimantos, S. Valentin, F. Wamser, B. Zeidler, M. Seufert, F. Loh, and P. Tran-Gia, "A Public Dataset for YouTube's Mobile Streaming Client," in *Proc. IEEE TMA*, Jun. 2018.

[175] developer.android.com. (2017) Android debugging tool (adb)-Android SDK Platform. Manual page. [Online]. Available: https://developer.android.com/studio/command-line/adb.html

[176] A. N. Kuznetsov. (2001) Linux traffic configuration tool (tc). Manual page. [Online]. Available: https://linux.die.net/man/8/tc

[177] developer.android.com. (2017) Android UI Automator-Android SDK Platform. Manual page. [Online]. Available: https://developer.android.com/training/testing/ui-automator.html

[178] C. Kreuzberger et al., "A Comparative Study of DASH Representation Sets Using Real User Characteristics," in *Proc. ACM NOSSDAV*, 2016.

[179] B. Foundation. (2013, Jun.) Tears of steel. Video. [Online]. Available: https://www.youtube.com/watch?v=OHOpb2fS-cM

[180] N. I. of Fundamental Studies Sri Lanka. (2016, Jan.) Science Copies Nature's Secrets. Video. [Online]. Available: https://www.youtube.com/watch?v=2d1VrCvdzbY

[181] Z. D. Fernsehen. (2017, Jul.) Heute Show. Video. [Online]. Available: https://www.youtube.com/watch?v=N2sCbtodGMI

[182] T. Karagkioules et al. (2018) A Public Dataset for YouTube's Mobile Streaming Client. Open dataset. [Online]. Available: http://qoecube.informatik.uni-wuerzburg.de/

[183] The Open Group, "IEEE Standard for Information Technology - Portable Operating System Interface (POSIX(R))," *IEEE Std 1003.1, 2004 Edition The Open Group Technical Standard*, Dec. 2008.

[184] N.N., "Content Provider Technical Requirements for Binge On," T-Mobile USA, Tech. Rep., Mar. 2016. [Online]. Available: https://www.t-mobile.com/content/dam/tmo/en-g/pdf/BingeOn-Video-Technical-Criteria-March-2016.pdf

[185] D. D. Vleeschauwer, H. Viswanathan, A. Beck, S. Benno, G. Li, and R. Miller, "Optimization of HTTP adaptive streaming over mobile cellular networks," in *Proc. IEEE INFOCOM*, Apr. 2013, pp. 898–997.

[186] A. Molavi Kakhki, F. Li, D. Choffnes, E. Katz-Bassett, and A. Mislove, "BingeOn Under the Microscope: Understanding T-Mobile's Zero-Rating Implementation," in *Proc. ACM Internet-QoE*, Aug. 2016, pp. 43–48.

[187] Y. Freund and R. E. Schapire, "A Decision-Theoretic Generalization of On-Line Learning and an Application to Boosting," *Journal of Computer and System Sciences*, vol. 55, no. 1, pp. 119–139, Aug. 1997.

[188] L. Breiman, "Random Forests," *Machine Learning*, vol. 45, no. 1, pp. 5–32, Oct. 2001.

[189] C. Seiffert, T. M. Khoshgoftaar, J. V. Hulse, and A. Napolitano, "RUSBoost: A Hybrid Approach to Alleviating Class Imbalance," *IEEE Transactions on Systems, Man, and Cybernetics - Part A: Systems and Humans*, vol. 40, no. 1, pp. 185–197, Jan. 2010.

[190] YouTube. (2015, Oct.) YouTube RED. [Online]. Available: https://www.youtube.com/red

[191] F. Loh, F. Wamser, C. Moldovan, B. Zeidler, T. Hoßfeld, D. Tsilimantos, and S. Valentin, "From Click to Playback: A Dataset to Study the Response Time of Mobile YouTube," in *Proceedings of the 10th ACM Multimedia Systems Conference*, ser. MMSys '19. New York, NY, USA: ACM, 2019, pp. 267–272. [Online]. Available: http://doi.acm.org/10.1145/3304109.3325819

[192] C. Timmerer and A. Bertoni, "Advanced Transport Options for the Dynamic Adaptive Streaming over HTTP," *CoRR*, vol. abs/1606.00264, 2016. [Online]. Available: http://arxiv.org/abs/1606.00264

[193] IAB and G. Camarillo, "Peer-to-Peer (P2P) Architecture: Definition, Taxonomies, Examples, and Applicability," RFC 5694, Nov. 2009. [Online]. Available: https://rfc-editor.org/rfc/rfc5694.txt

[194] D. Stutzbach and R. Rejaie, "Understanding Churn in Peer-to-peer Networks," in *Proceedings of the 6th ACM SIGCOMM Conference on Internet Measurement*, ser. IMC '06. New York, NY, USA: ACM, 2006, pp. 189–202. [Online]. Available: http://doi.acm.org/10.1145/1177080.1177105

[195] L. Alkwai and A. Gazdar, "Dynamic quality adaptive P2P streaming system," in *2016 International Wireless Communications and Mobile Computing Conference (IWCMC)*, Sep. 2016, pp. 158–163.

[196] L. Natali and M. L. Merani, "Successfully Mapping DASH Over a P2P Live Streaming Architecture," *IEEE Transactions on Circuits and Systems for Video Technology*, vol. 27, no. 6, pp. 1326–1339, June 2017.

[197] F. Qian, L. Ji, B. Han, and V. Gopalakrishnan, "Optimizing 360 Video Delivery over Cellular Networks," in *Proceedings of the 5th Workshop*

on All Things Cellular: Operations, Applications and Challenges, ser. ATC '16. New York, NY, USA: ACM, 2016, pp. 1–6. [Online]. Available: http://doi.acm.org/10.1145/2980055.2980056

[198] J. Le Feuvre and C. Concolato, "Tiled-based Adaptive Streaming Using MPEG-DASH," in *Proceedings of the 7th International Conference on Multimedia Systems*, ser. MMSys '16. New York, NY, USA: ACM, 2016, pp. 41:1–41:3. [Online]. Available: http://doi.acm.org/10.1145/2910017.2910641

[199] O. A. Niamut, E. Thomas, L. D'Acunto, C. Concolato, F. Denoual, and S. Y. Lim, "MPEG DASH SRD: Spatial Relationship Description," in *Proceedings of the 7th International Conference on Multimedia Systems*, ser. MMSys '16. New York, NY, USA: ACM, 2016, pp. 5:1–5:8. [Online]. Available: http://doi.acm.org/10.1145/2910017.2910606

[200] M. Ben Yahia, Y. Le Louedec, G. Simon, and L. Nuaymi, "HTTP/2-Based Streaming Solutions for Tiled Omnidirectional Videos," in *2018 IEEE International Symposium on Multimedia (ISM)*, Dec 2018, pp. 89–96.

[201] B. Rainer and C. Timmerer, "A Generic Utility Model Representing the Quality of Sensory Experience," *ACM Trans. Multimedia Comput. Commun. Appl.*, vol. 11, no. 1s, pp. 14:1–14:17, Oct. 2014. [Online]. Available: http://doi.acm.org/10.1145/2648429

[202] P. Reichl, S. Egger, S. Mller, K. Kilkki, M. Fiedler, T. Hoßfeld, C. Tsiaras, and A. Asrese, "Towards a comprehensive framework for QOE and user behavior modelling," in *2015 Seventh International Workshop on Quality of Multimedia Experience (QoMEX)*, May 2015, pp. 1–6.

[203] H. He, H. Shan, A. Huang, and L. Sun, "Resource Allocation for Video Streaming in Heterogeneous Cognitive Vehicular Networks," *IEEE Transactions on Vehicular Technology*, vol. 65, no. 10, pp. 7917–7930, Oct 2016.

[204] X. Foukas, G. Patounas, A. Elmokashfi, and M. K. Marina, "Network Slicing in 5G: Survey and Challenges," *IEEE Communications Magazine*, vol. 55, no. 5, pp. 94–100, May 2017.

Titre : Optimisation de la vidéo adaptative streaming sur les réseaux mobiles

Mots clés : diffusion vidéo, resaux mobile, adaptation

Résumé : Grâce à l'évolution des technologies des réseaux et des dispositifs mobiles (téléphones, tablettes), les utilisateurs sont amenés à utiliser de plus en plus de services a forte consommation de bande passante, comme le streaming vidéo. Selon un report de CISCO, en 2017, le streaming vidéo mobile représentait 58% du trafic de données mobile mondial, un pourcentage qui devrait atteindre 79% d'ici 2022. Ce trafic est principalement constitué de vidéo à la demande (VoD) sur HTTP Adaptive Streaming (HAS), qui est désormais une technologie incontournable des dispositifs mobiles. Les solutions HAS utilisent des algorithmes d'adaptation de débit, qui ajustent de manière transparente le débit du flux multimédia pour compenser les conditions changeantes du réseau.

Néanmoins, la transmission vidéo sur les réseaux mobiles fait toujours face à des défis importants car ces réseaux se caractérisent par une variation continuelle du débit, principalement attribuable aux effets physiques associés à la propagation radio ainsi qu'à la mobilité des utilisateurs (interruptions de session).

L'objectif de cette thèse est de traiter ces défis de la diffusion vidéo dans les réseaux mobiles et de proposer des solutions optimisées sous deux perspectives distinctes. Dans la perspective des fournisseurs de services vidéo Over the Top (OTT), nous proposons de nouveaux algorithmes d'adaptation du débit mobile; tandis que, dans la perspective des vendeurs d'équipements de télécommunication et des opérateurs réseau, nous explorons les avantages que la collecte du trafic HAS et de son analyse peuvent apporter à la gestion du réseau.

Plus précisément, lors de l'évaluation des performances de systèmes HAS actuels, nous avons identifié 3 problèmes qui nécessitent des nouvelles et meilleurs solutions que l'état de l'art.

Le premier problème concerne les données d'entrée qui pilotent les algorithmes d'adaptation de débit. Bien que plusieurs algorithmes d'adaptation de débit aient été introduits pour améliorer la qualité d'expérience (QoE) de l'utilisateur, seuls quelques-uns exploitent les informations provenant des couches inférieures et des capteurs actuellement disponibles sur tous les appareils mobiles modernes. Au contraire, la plupart des techniques d'adaptation de débit reposent sur une estimation de ce dernier ou sur d'autres mesures disponibles à niveau application, telles que la quantité de données pré-chargées. Dans cette direction, nous proposons une nouvelle solution d'adaptation de débit basée sur le contexte et intégrant les informations des capteurs du téléphone.

Le deuxième problème concerne les fournisseurs de services vidéo OTT. La plupart des algorithmes d'adaptation de débit existants dépendent de modèles statistiques et se heurtent donc à des difficultés quand il est nécessaire de généraliser de manière appropriée ces modèles en dehors des scénarios prévus lors de la modélisation statistique. Pour pallier cette limitation, nous proposons un nouvel algorithme d'adaptation du débit basé sur l'apprentissage en ligne, qui fonctionne bien sur un large éventail de scénarios de streaming. Cela ne nécessite aucun réglage de paramètre, aucune modification en fonction du type d'application ou des hypothèses statistiques pour le canal.

Enfin, le troisième problème concerne l'absence de jeux de données disponibles et actualisés pour le trafic du streaming vidéo mobile. Afin de mieux comprendre le trafic HAS et d'obtenir des données fiables qui permettraient aux opérateurs d'optimiser leurs réseaux, nous avons mené une vaste campagne expérimentale visant à collecter des informations sur plusieurs couches à partir du trafic en streaming. L'ensemble de données qui en résulte est mis à la disposition de la communauté académique et industrielle, les données étant enregistrées aux niveaux réseau, transport et application. Nous avons aussi enregistré les données du streaming vidéo sur QUIC, pour la première fois. Enfin, nous avons utilisé ces données pour concevoir une nouvelle technique de profilage de trafic, basée sur l'apprentissage automatique, et qui fournit une estimation des parametres du système HAS à partir de mesures "passives" uniquement.

Title : Optimization of Adaptive Video Streaming in Mobile Networks

Keywords : Mobile networks, adaptive streaming, optimization

Abstract : As mobile networking technology is experiencing perpetual evolution and the computing capabilities of mobile devices are being constantly enhanced, the demand for bandwidth-intensive mobile multimedia consumption is currently experiencing an unprecedented surge. In 2017, mobile video streaming accounted for 58% of the global mobile data traffic, a percentage that is projected to reach a striking 79% by 2022. Most of this traffic is Video on Demand (VoD) over HTTP Adaptive Streaming (HAS), which undoubtedly becomes fast an integral part of the mobile client's life. HAS solutions employ rate adaptation algorithms, that seamlessly adjust the rate of the media stream, to compensate for changing network conditions. Most notably, Dynamic Adaptive Streaming over HTTP (DASH) is a standard for HAS, that uses existing web server infrastructure and is now the dominant solution for video delivery.

Nonetheless, video delivery over mobile networks still faces substantial challenges, primarily in mobile networks, that are commonly characterized by throughput variation; primarily attributed to physical effects associated with radio propagation, along with short-term session interruptions attributed to user mobility. Depending on the current user location, channel degradation can have a detrimental impact on user Quality of Experience (QoE).

The scope of this dissertation is to treat these challenges of video delivery in mobile networks and expedite its optimization, under two distinct perspectives. Under the perspective of Over the Top (OTT) video service providers, we propose new mobile rate adaptation algorithms; whereas, under the perspective of telecommunication equipment vendors and network service operators, we explore the merits of HAS traffic collection and its analysis, on network management. Upon conducting a performance evaluation of state-of-the-art HAS schemes, we have identified 3 open issues in the current state of HAS.

The first issue concerns the type of input that drives the rate adaptation logic. Although several rate adaptation algorithms have been introduced in order to improve user QoE, only few leverage cross-layer and sensor information that is nowadays readily available in all modern mobile devices, while most rely on either throughput estimation or application-level readings, such as the amount of pre-fetched data. In that direction, we propose a new context-aware rate adaptation solution, that incorporates cellular sensor information into the rate adaptation process.

The second open issue concerns OTT video service providers, who are continuously expanding their services to include more diverse user classes, network scenarios, and streaming applications. Most existing rate adaptation algorithms depend on statistical models for the unknowns and thus face complications at generalizing appropriately well beyond a certain scope of usage. To mitigate this limitation, we propose a novel rate adaptation algorithm based on online learning, that performs well over a wide spectrum of streaming scenarios due to its design principle; its ability to learn. It does so without requiring any parameter tuning, modifications according to application type or statistical assumptions for the channel.

Last, the third open issue regards the absence of available data-sets for up-to-date mobile video streaming traffic. To better understand HAS traffic and also in order to obtain reliable data that would ultimately enable operators to optimize their networks, we conduct an extensive experimental campaign to collect cross-layer information from streaming traffic. The resulting data-set is made publicly available, with data recorded at the transport, network and application layer; capturing video streaming traffic over QUIC, for the first time. Additionally, we use this data-set to design a novel traffic profiling solution, based on machine learning, that estimates parameters of HAS applications from passive measurements.

Institut Polytechnique de Paris
91120 Palaiseau, France