Julius-Maximilians-Universität Würzburg
Institut für Informatik
Lehrstuhl für Kommunikationsnetze
Prof. Dr. Tobias Hoßfeld

Performance Modeling of Mobile Video Streaming

Christian Moldovan

Würzburger Beiträge zur
Leistungsbewertung Verteilter Systeme

Bericht 1/20

Würzburger Beiträge zur Leistungsbewertung Verteilter Systeme

Herausgeber

Prof. Dr. Tobias Hoßfeld, Prof. Dr.-Ing. P. Tran-Gia
Universität Würzburg
Institut für Informatik
Lehrstuhl für Kommunikationsnetze
Am Hubland
D-97074 Würzburg

Tel.: +49-931-31-86631
Fax.: +49-931-31-86632
email: tobias.hossfeld@uni-wuerzburg.de

Satz

Reproduktionsfähige Vorlage des Autors.
Gesetzt in LaTeX Linux Libertine 10pt.

ISSN 1432-8801

Performance Modeling of Mobile Video Streaming

Dissertation zur Erlangung des
naturwissenschaftlichen Doktorgrades
der Julius–Maximilians–Universität Würzburg

vorgelegt von

Christian Moldovan

geboren in

Temeschburg

Würzburg 2020

Eingereicht am: 4.8.2020
bei der Fakultät für Mathematik und Informatik
1. Gutachter: Prof. Dr. Tobias Hoßfeld
2. Gutachterin: Izv. Prof. Dr. Sc. Lea Skorin-Kapov
Tag der mündlichen Prüfung: 21.10.2020

Danksagung

Bei der Erstellung der vorliegenden Arbeit habe ich viel Unterstützung und Hilfe erfahren. All denen, die zum guten Gelingen beigetragen haben, möchte ich hier meinen Dank aussprechen. Zu allererst danke ich meinem Doktorvater Prof. Tobias Hoßfeld, der es mir ermöglicht hat, zu promovieren. Nicht zuletzt durch seinen persönlichen Einsatz, seine Fähigkeiten wissenschaftliches Arbeiten zu vermitteln und seine vielfältigen internationalen Kontakte sorgte er für herausragendes Umfeld, in dem ich die Forschungsarbeiten für meine Doktorarbeit durchführen konnte. Zusätzlich bot er mir Gelegenheit, an mehreren Forschungsprojekten teilzunehmen und hochrangige Fachkonferenzen auf der ganzen Welt zu besuchen, wo ich meine Ideen mit anderen Forschern diskutieren konnte. Rückblickend war dies eine der wesentlichen Grundlagen für die Ergebnisse, die ich in meiner Forschung erzielen konnte. Darüber hinaus gab er mir die Möglichkeit, immer wieder eigene Ideen zu entwickeln und in Projekten auch selbst Verantwortung zu übernehmen. Diese außerordentlichen Rahmenbedingungen haben entscheidend zum Gelingen meiner Arbeit beigetragen und dafür bin ich Prof. Hoßfeld sehr dankbar. Bedanken möchte ich mich auch bei Prof. Alexander Wolff, dessen Betreuung meiner Bachelorarbeit mir dabei geholfen hat, ein Verständnis für Optimierung und lineare Programmierung zu entwickeln, was die Grundlage eines Kapitels dieser Arbeit war. Weiterhin fungierte er als Prüfer bei meiner Disputation. Für diese Unterstützung danke ich ihm herzlich. Weiterhin gebührt mein Dank Prof. Rainer Kolla, der den Vorsitz der Prüfungskommission übernahm Dankbar bin ich auch Prof. Lea Skorin-Kapov, die das zweite Gutachten erstellt hat und Prof. Paul E. Heegaard. In zahlreichen gemeinsamen Veröffentlichungen konnte ich viel

von ihnen über das wissenschaftliche Arbeiten lernen und von ihren Erfahrungen profitieren. Insbesondere durch die gute Betreuung meiner Masterarbeit ermöglichten mir Dr. Burger, Dr. Wamser und Dr. Lehrieder einen erfolgreichen Start in meine Arbeit als wissenschaftlicher Mitarbeiter. In vielfältiger Weise haben meine Kollegen an der UDE und der JMU zum erfolgreichen Abschluss dieser Arbeit beigetragen, sei es in fachlichen Diskussionen, durch wertvolle Kommentare zu wissenschaftlichen Arbeiten oder auch durch technische Unterstützung. Ohne alle einzeln beim Namen zu nennen möchte ich mich herzlich bei ihnen für die angenehme Atmosphäre und die gute Zusammenarbeit bedanken. Zusätzlich haben sich an den Lehrstühlen, an denen ich in den vergangen Jahren tätig war viele Freundschaften entwickelt und ich denke gerne an die zahlreichen Unternehmungen zurück, die ich mit ihnen auch außerhalb des Lehrstuhls durchgeführt habe. Im Rahmen von Übungen, Vorlesungen und Abschlussarbeiten eine große Zahl von Studierenden betreut. Mit ihrem Engagement und ihren Ideen haben sie dafür gesorgt, dass ich an der Betreuung nicht nur viel Freude hatte sondern von ihr auch fachlich stark profitieren konnte. Für die tatkräftige Hilfe in allen administrativen Tätigkeiten bin ich Frau Quadt und Frau Wichmann sehr dankbar. Abschließend möchte ich mich herzlich bei meiner Familie, meiner Partnerin Jana und meiner Tochter Edda bedanken. Ohne den Rückhalt und die vielfältige Unterstützung wäre diese Doktorarbeit nicht möglich gewesen.

Contents

1 Introduction

In the past two decades, there has been a trend to move from traditional television to Internet-based video services. With video streaming becoming one of the most popular applications in the Internet and the current state of the art in media consumption, quality expectations of consumers are increasing. Low quality videos are no longer considered acceptable in contrast to some years ago due to the increased sizes and resolution of devices [24]. If the high expectations of the users are not met and a video is delivered in poor quality, they often abandon the service [25]. Therefore, Internet Service Providers (ISPs) and video service providers are facing the challenge of providing seamless multimedia delivery in high quality [26]. Currently, during peak hours, video streaming causes almost 58% of the downstream traffic on the Internet [27]. With higher mobile bandwidth, mobile video streaming has also become commonplace. According to the 2019 Cisco Visual Networking Index [28], in 2022 79% of mobile traffic will be video traffic and, according to Ericsson, by 2025 video is forecasted to make up 76% of total Internet traffic [29]. Ericsson further predicts that in 2024 over 1.4 billion devices will be subscribed to 5G [30], which will offer a downlink data rate of 100 Mbit/s in dense urban environments [31].

One of the most important goals of ISPs and video service providers is for their users to have a high Quality of Experience (QoE). The QoE describes the degree of delight or annoyance a user experiences when using a service or application [32]. In video streaming the QoE depends on how seamless a video is played and whether there are stalling events or quality degradations [33, 34]. These characteristics of a transmitted video are described as the application layer Quality of Service (QoS) [35]. In general, the QoS is defined as "the totality

of characteristics of a telecommunications service that bear on its ability to satisfy stated and implied needs of the user of the service" by the ITU [36]. The network layer QoS describes the performance of the network and is decisive for the application layer QoS.

Another goal of service providers is to keep their customers engaged with their services. User Engagement has affective, behavioral, and cognitive aspects [37]. It can be objectively described by different metrics, such as the frequency and the duration of user interactions with an application [38]. It can also be described by the user involvement and participation [39]. In video streaming, poor QoS typically leads to poor User Engagement which leads to users abandoning the video [25].

In Internet video, typically a buffer is used to store downloaded video segments to compensate for network fluctuations. If the buffer runs empty, stalling occurs. If the available bandwidth decreases temporarily, the video can still be played out from the buffer without interruption. There are different policies and parameters that determine how large the buffer is, at what buffer level to start the video, and at what buffer level to resume playout after stalling. These have to be finely tuned to achieve the highest QoE for the user. If the bandwidth decreases for a longer time period, a limited buffer will deplete and stalling can not be avoided. An important research question is how to configure the buffer optimally for different users and situations. In this work, we tackle this question using analytic models and measurement studies. With HTTP Adaptive Streaming (HAS), the video players have the capability to adapt the video bit rate at the client side according to the available network capacity. This way the depletion of the video buffer and thus stalling can be avoided. In HAS, the quality in which the video is played and the number of quality switches also has an impact on the QoE [40–42]. Thus, an important problem is the adaptation of video streaming so that these parameters are optimized. In a shared WiFi multiple video users share a single bottleneck link and compete for bandwidth. In such a scenario, it is important that resources are allocated to users in a way that all can have a similar QoE [43]. In this work, we therefore investigate the possible fairness gain when moving from

network fairness towards application-layer QoS fairness. In mobile scenarios, the energy and data consumption of the user device are limited resources and they must be managed besides the QoE. Therefore, it is also necessary, to investigate solutions, that conserve these resources in mobile devices. But how can resources be conserved without sacrificing application layer QoS? As an example for such a solution, this work presents a new probabilistic adaptation algorithm that uses abandonment statistics for ts decision making, aiming at minimizing the resource consumption while maintaining high QoS.

With current protocol developments such as 5G, bandwidths are increasing, latencies are decreasing and networks are becoming more stable, leading to higher QoS [44]. This allows for new real time data intensive applications such as cloud gaming, virtual reality and augmented reality applications to become feasible on mobile devices which pose completely new research questions. The high energy consumption of such applications still remains an issue as the energy capacity of devices is currently not increasing as quickly as the available data rates. In this work we compare the optimal performance of different strategies for adaptive 360° video streaming.

1.1 Scientific Contribution

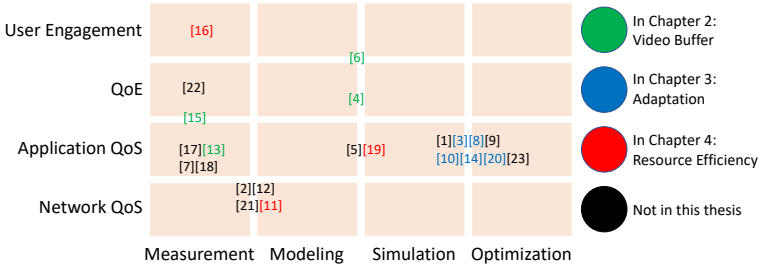


Figure 1.1: Contribution of this work illustrated by a cartography of the research studies carried out by the author.

The following paragraph lists the scientific contributions and assigns them to the methodologies used in this monograph. A classification of these contributions is given in Figure 1.1. On the horizontal axis, the methodology is depicted while the vertical axis shows the topic or layer on which we conduct the investigation. The main topic of this work is the performance evaluation of adaptive video streaming, which can be tackled from different angles with different methods. Our investigation includes measurement studies, simulations, optimization techniques, and analytical methods including queueing models. A graphical outline of this work is given in Figure 1.2.

In the first chapter, we investigate the buffer in which video segments are stored temporarily until playout to compensate for network fluctuations. As a first contribution, we combine two QoE models with different impact factors to create a new unified QoE model [4]. We propose an $M/M/1$ queueing model for investigating the buffer in video streaming under various constraints and derive key performance indicators (KPI) for different situations [4]. For this model we present results on the impact of the buffer size and the offered load of the network on the QoE using a new QoE model. Furthermore, we conduct a measurement study on adaptive streaming in which we investigate the impact of a further buffer parameter on the QoE [13, 15]. We extend the queueing model to a more general $M/G/1$ model for which we derive the KPIs for three different buffer policies [6]. Furthermore, we propose a stalling ratio-based model for the user engagement in video streaming [6]. This allows us to predict how long users will engage with content based on stalling.

In the second chapter, we consider ways to optimize the QoS of adaptive streaming. We model the buffer of video streaming for a generic buffer-based adaptation with a Markov model which allows us to derive key impact factors for video streaming analytically. Furthermore, we use it to conduct an analysis of the impact of available quality layers on the stalling rate. As the main part of this chapter, we present a group of optimization problems for adaptive streaming and corresponding linear and quadratic programs [8, 3, 10, 14]. In these problems the goal is to maximize the video quality, fairness and to minimize quality switches

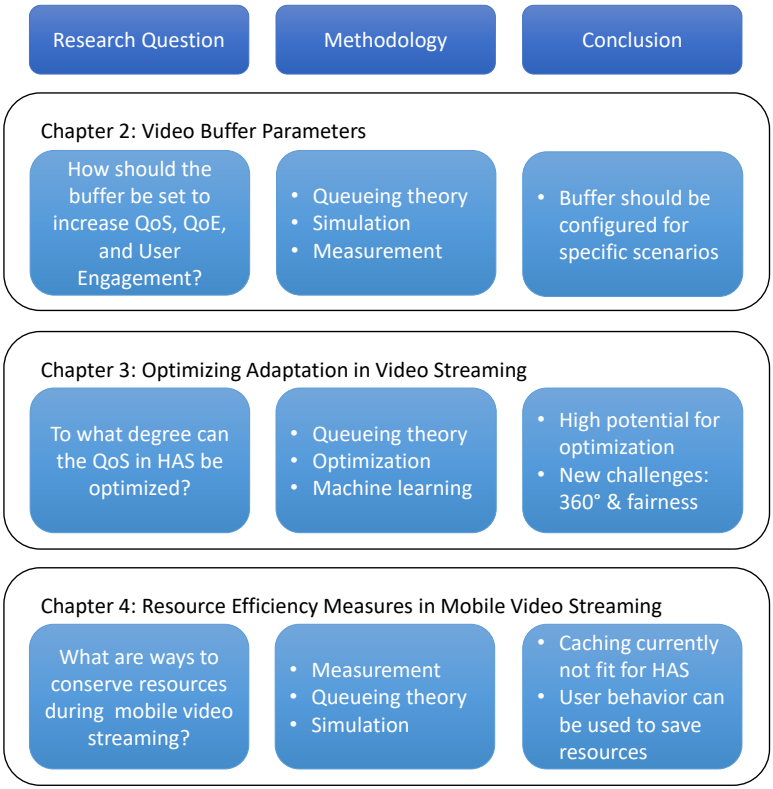


Figure 1.2: Structure of this monograph.

and to avoid stalling. We use such a problem to compare the optimal performance of different approaches to viewport prediction on the QoS in 360° video streaming [20]. The presented programs can be used for the development of new adaptation heuristics, e.g., by comparing how large the performance gap is compared to the optimal solution.

In the third chapter, we shift the focus away from the QoS of video streaming to resource consumption and conservation. We conduct a field study with public WiFi caches from which we measure the typical traffic mix at WiFi hotspots [11]. This gives us an indication to what degree traffic could be cached at WiFi hotspots. Furthermore, we present the results of a five-year study on mobile users and their viewing habits and behavior [16]. Data on the user behavior can then be used to improve the resource efficiency of video streaming. We present a novel energy and data efficient adaptation algorithm for mobile networks that uses abandonment statistics [19]. This means that users can enjoy their videos for a longer period, before the battery or their mobile data is consumed.

1.2 Outline of this Thesis

The remainder of this monograph is structured as follows. In Chapter 2, we investigate the buffer in video streaming. We start with giving an overview on background and related work on general topics that are discussed throughout this monograph. Using a queueing model, we first analyze how the buffer size impacts the QoE. This is done, considering the user preferences regarding stalling. In a measurement study we observe the impact of changing the maximum buffer size on the QoE. Finally, we study three buffering strategies using analytical and numerical methods.

In Chapter 3, we optimize different aspects of adaptive video streaming. At first, we discuss background and publications related to this chapter. We then present a queueing model for adaptive video streaming to derive key impact parameters. Furthermore, we use it to determine the impact of the number of quality representations in which the video is offered on stalling. Next, we

propose various linear and quadratic programs that maximize quality and fairness, minimize the number of switches and avoid stalling. Finally, we investigate different approaches to viewport-prediction in 360-degree videos which we also optimize using linear programming.

In Chapter 4, we look at data- and energy-efficient mechanisms for mobile video streaming. As a first step, User Engagement and resource efficiency in video streaming is discussed. We then analyze the behavior of mobile video users. We then develop a new data- and energy-efficient adaptation algorithm that uses viewer abandonment statistics and adapts to the cellular network technology. In a comparison with another algorithm we find that it is much more resource-efficient while similar in terms of QoS. As a third step, we conduct a field study in which we deploy WiFi caches at public locations and investigate the traffic mix. We complete this with an analysis of the cache performance.

Chapter 5 summarizes this work and draws conclusions.

2 Video Buffer Parameters

In online video streaming, the buffer is the client-side memory into which video segments are downloaded and temporarily stored. It is used for the purpose of storing data in order to compensate for short-term fluctuations in both downloading and video coding. It is used precisely to overcome the instability of the network in the event of temporary downtime, stagnation or fluctuating issues throughput. Without the buffer, network errors would immediately impair the playback quality, which is not in favor of the user. After downloading, the player plays a segment from the buffer. It is removed from the buffer, and subsequently the next segment is played. If the segments are played faster than they are downloaded, the buffer runs empty and stalling occurs. As soon as the buffer is filled sufficiently, the playout is resumed. If the buffer is short, there is a greater risk that the buffer runs empty and stalling occurs eventually. If the buffer is large, more data is downloaded unnecessarily if the user abandons the video early. Furthermore, the delay until the video initially starts is greater, if the user must wait longer to fill a large buffer. Therefore, it is important to find a balanced buffer size that benefits the individual user or the specific scenario. For example, an impatient user who is browsing through videos may prefer a short buffer for a reduced initial startup delay, while a patient user who watches a long movie would prefer a larger buffer to avoid video degradation. The impact that a degradation has on the QoE does not only depend on the user, but also on the type of content (e.g., live, video on demand (VOD)), the video content and the usage scenario.

Most video players use HAS to avoid stalling as long as the lowest bit rate representation of the video can be played with the current bandwidth. In mobile

scenarios however, this is often not sufficient. For example, on amazon video the lowest available video bit rate lies at around 900 kbit/s as of 2020 which is not available in many regions. Therefore, it makes sense to ignore adaptation mechanisms when assuming low bandwidth scenarios to simplify the analysis of video streaming models. In such scenarios our results not only apply to non-adaptive streaming, but also to HAS. Regarding the QoE in video streaming, we investigate the impact of the buffer on three types of impairments: the frequency of stalling events, the duration of stalling events, and the initial delay. All three can be controlled through the buffer to some extent. Furthermore, users have different viewing habits and consume content in different ways. We note that video sessions can be divided into three usage scenarios:

Watch Later In this scenario, the user is interested in watching a longer piece of content such as a movie. For him or her, it is most important, that the video is transmitted in high quality and without interruptions. In this scenario, we assume a patient user and it is not necessary that the video starts immediately. We can solve this issue by having a large buffer that is filled slowly with high quality content. Playback starts as soon as the buffer is sufficiently filled, for example after 5 minutes.

Regular Video Streaming In the default video streaming scenario, the user wants to watch a video immediately. In this case, a shorter buffer is required for a good trade-of between a low initial delay and some stalling.

Video Browsing In a video browsing scenario the user is searching or browsing for a certain piece of content that he or she is interested in. This process can be within a video platform, across different video platforms or by skipping ahead within a single video in search for a specific scene. Since the user only watches a few seconds of video content before switching to the next video, a small buffer is sufficient. This way, the delay between videos is kept short.

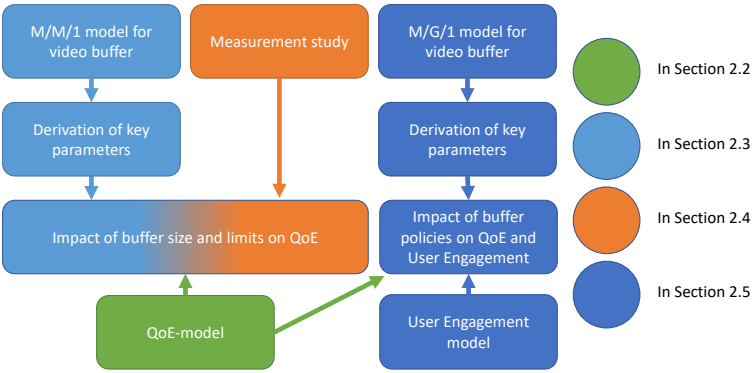


Figure 2.1: Structure of the contributions of this chapter.

In this chapter, we investigate video streaming from different perspectives with the following research question in mind: *How does the buffer affect the application layer QoS and the QoE in HTTP Video Streaming?*

We answer this question by conducting performance evaluation using queueing models and with a measurement study using wireless network traces. To understand the impact of the buffer better, we focus on the following parameters: the maximum buffer size, the buffer threshold when video playout starts, and various buffer policies. For the main performance metric we use a configurable QoE model that is configured with user parameters to model the QoE for three usage scenarios. Furthermore, we consider the play time as a performance metric for the User Engagement and investigate how it relates to the QoE. The play time is defined as the time that a user spends watching a video before he abandons it.

This chapter is based on content published on different conferences and workshops [4, 6, 13, 15]. An overview of the structure of the contributions of this chapter is given in Figure 2.1. In the next section, we will present related work on QoS, QoE and User Engagement in HAS. In Section 2.2, we present a multiplicative and an additive QoE model that include the length and the duration of

stalling events and the initial delay, which is based on [4]. Section 2.4 uses these QoE models to conduct a practical evaluation of the buffer size in HAS, which is based on [13] and [15]. We then investigate the impact of buffer policies on the QoE and the User Engagement which is based on [6]. Finally, we discuss lessons learned.

2.1 Background and Related Work

In this section we discuss related work on video streaming. We start by explaining how the network QoS has an impact on the application layer QoS which in return influences the user's QoE. We then describe how adaptation algorithms aim to solve issues in low bandwidth scenarios by trading off low impact characteristics for high impact characteristics.

2.1.1 QoE in Video Streaming

Service providers compete to provide the best QoE for users. The QoE is particularly important in video streaming. In order to find out how the QoE can be optimized, user studies investigate the impact of video degradation effects on the QoE [45]. Typically, the subjective opinion of users is aggregated to a single value, such as the Mean Opinion Score (MOS) which is examined in detail in [46]. However, it is also possible to use quantiles [47], distributions [47, 48] or the Standard deviation of Opinion Scores [49].

QoE in video streaming is impacted by the application QoS which depends on the network QoS [45]. In user studies, it was found that stalling has a higher impact on the QoE than the initial delay of a video [33, 50] or a reduction in the frame rate [51]. Furthermore, stalling at irregular intervals is worse than periodic stalling [51]. Another user study determined that the QoE can be improved by estimating the best initial bit rate of a video [52]. For practical purposes, this is a great addition for any adaptation heuristic.

In image processing, quantization is a lossy compression technique where the data volume of an image is reduced while information is lost. This can result in artifacts and other distortion effects. The degree of saved data and lost information can be controlled by a quantization parameter. Using subjective subjective studies, the authors of [53] find that users are more sensitive to stalling than to an increase of a quantization parameter in the video encoder, especially for lower values of the quantization parameter. In [54], the impact of network and application parameters on stalling is analyzed. In addition, the impact of stalling on QoE is investigated in a subjective user study in this section.

According to [55], the transport protocol has an impact on the application layer QoS. For example, having insufficient bandwidth may lead to stalling when using TCP, while it would lead to a quality degradation when using UDP. This is a reason why most video hosting services transmit their videos via TCP. Even live-streaming platforms such as Twitch use TCP instead of UDP. A great overview of QoE in adaptive video streaming is also given in [56].

Models that describe the relation between the QoS and the QoE can be used for QoE prediction and QoE management. There exist also some multi-factor QoE models for adaptive streaming. For example, such a model is provided in [57]. The authors propose a model, that includes the initial delay, varying perceptual quality, and interruptions. The authors of [58] use results from a subjective study to create an objective model for QoE that accounts for quality variations and stalling events. A comparison of different QoE models with subjective test is conducted in [33]. Additive and Multiplicative multi-factor QoE models are investigated in [59, 60].

The instability metric is also sometimes considered for the evaluation of QoS in video streaming [61]. It is the fraction of successive chunk requests by a player in which the requested bit rate does not remain constant. For example, in [61], it is defined as the weighted sum of all switch steps observed within the last $k = 20$ seconds divided by the weighted sum of bit rates in the last 20 seconds.

From these works, we conclude that the key impact parameters for QoE in adaptive video streaming are the video quality, the frequency of layer switches,

the initial delay, and the frequency and length of stalling events. All these parameters can be determined easily in the model presented in this chapter.

2.1.2 Adaptive Video Streaming

HAS is the de-facto standard [62] of delivering video contents of prevailing video platforms in the Internet like YouTube, Netflix or Amazon Prime. HAS allows the client to adapt the video bit rate according to the available network bandwidth. The key idea is to overcome stalling, as the video interruptions have a major impact on the Quality of Experience [63] compared with quality switches or initial delays before the video starts to play out [33, 50]. To this end, the video client measures relevant parameters and requests the next part of the video in an appropriate bit rate. This selection is done by a heuristic or adaptation strategy that aims to avoid stalling while playing the video on the highest possible video quality. Therefore, the video must be available in multiple bit rates, i.e., in different quality levels / layers (e.g. different resolutions) and split into small segments each containing a few seconds of playtime. The number and choice of layers provided has a significant impact on the user's satisfaction [64].

For the video quality level decision, the client typically measures the current bandwidth or the buffer status. In this context, the adaptation strategy is referred to as bandwidth-based and buffer-based strategy, respectively. A large evaluation of proposed mechanisms can be found in [65], [66], and [67]. An overview on HAS technology, the prevailing protocols and mechanisms, as well as implementations are presented in [56].

2.1.3 User Engagement in Video Streaming

The authors of [25] measure QoE-metrics and User Engagement from various sites, different types of content (short VOD, long VOD, and live video), and also distinguish other kinds of parameters. Their results show that a high buffering ratio lowers User Engagement, with the impact being stronger for short videos. Similarly, a high bit rate has a significant impact in the live scenario while it does

not in VOD. In [68] traffic during a single live event is measured and the impact of QoE metrics on User Engagement is analyzed. Their results show that the buffering ratio and the bit rate have a high impact on User Engagement. Further, they noted that the video play time may depend on various other factors such as user behavior. A correlation between QoE and User Engagement was also recognized.

A paper [69] conducted a large-scale measurement study that looked at the abandonment rate — which can be another appropriate User Engagement metric — for mobile video streaming. Using data from the study, a model is proposed that can predict User Engagement in mobile video streaming with a high accuracy based on network statistics.

In two further publications, [70, 71] the authors measured User Engagement and video session quality and run machine learning algorithms on it. Through this effort, they highlight the challenges of obtaining a robust video QoE model from such metrics. In [72] the authors propose a machine learning based model that uses information on user interactions, such as pausing, seeking, and abandonment, during videos to predict important application layer QoS metrics. The possibility to predict QoS from user interactions shows how important such information can be for QoE management. The importance of the relationship between the user behavior and the QoE is also outlined in [73]. In that work, the authors focus on the impact of user impatience in shared systems and how waiting times impact the QoE. They find that user impact has a great impact on QoE but little impact on the probability that a user is served. And finally, a paper [74] puts viewer behavior in relation to video quality metrics. Of note is the observation that an increase in the initial delay of a video stream also directly leads to a higher abandonment rate.

User Engagement can be defined in many ways, e.g., time spent on a website, abandonment rate, interactions, click rate, attention paid, number of comments. It is interesting from the perspective of the content provider and the service provider since high User Engagement leads to a higher number of ad views or sales. For video streaming services, we need an easily measurable, objective

metric that describes how much content users consume and how willing they are to view ads. Therefore, we define User Engagement as the play time of a video. An overview of models for User Engagement metrics for a number of online services is given in [75]. In addition, users might abandon a service because of stalling, thereby reducing User Engagement. Using this definition, it seems plausible that video streaming platforms, content providers, or video service providers generate revenue based on User Engagement making it a critical metric.

2.1.4 QoE Management

So far, all these papers have looked at the significance of specific User Engagement metrics but lack in terms of mapping measured QoS values to a specific QoE and how to facilitate this information for network and QoE management aspects in light of the future Internet development. The following publications investigate this from an Internet Service Provider (ISP) point of view.

For an ISP it is generally more difficult to estimate the video streaming QoE in its network and may require invasive measures, such as Deep Packet Inspection (DPI). However, this is possible with the approach suggested in [50] which was also successfully deployed in the network of a large European mobile operator [76]. Data gained from such monitoring endeavors can be further utilized, e.g., to enable flow-based traffic management for improving QoE via SDN as presented in [77].

Once such influence factors from all network layers have been collected, they can be mapped to QoE according to existing QoS-QoE models and relationships. As this is not without challenges, [78] surveys current research activities on QoE management with a focus on wireless networks where QoE management has mostly just been considered in terms of resource scheduling and resource allocation decisions. Furthermore, the involved technology continuously advances and introduces new challenges. Such as the migration of services to the cloud [79], and cloud gaming [80, 81]. Home gateways are also a starting point to optimize QoE at a small scale. [82] shows that even with just very basic knowledge of the

user's service requirements, a significant improvement in QoE can be achieved through methods such as application prioritization and traffic shaping.

In contrast, managing QoE based on QoS estimates the user's QoE with objective metrics. For example, [83] defines a reception ratio as the ratio between download throughput and video encoding rate. For some ISPs this may already be sufficient to determine whether stalling occurs or not and how the user reacts in response. Reference [50] concludes that this ratio cannot be directly related to the QoE, yet it is still a good indicator if there are problems in the network. Both [68] and [84] investigate and review different engagement measures and how they are impacted by QoE metrics.

The problem with many such QoE management approaches is that for some services the models are not fully understood or there may be further, hidden influence factors which are not captured by the employed methodology, e.g., recency effects. Additionally, even if the models are well established, e.g. for HTTP video streaming, it may still be difficult to measure the related parameters. Similarly, looking at research efforts involving video streaming engagement, often the reasons are unclear why a user stops watching the video. It may stem from quality issues in the streaming process, but it may very well also just be that the user lost interest in that particular content.

Many QoE metrics are not directly measurable in an HTTPS environment since it is difficult to estimate video quality and stallings from encrypted traffic. Only recently, first successful attempts have been made to deduce the application layer QoS and the QoE from encrypted network traffic [21, 85, 86]. In contrast, the User Engagement can be measured more easily at large scale. Additionally, the effects of low quality will be directly visible in this User Engagement measures. Since there seems to be a lot of value in investigating the relation between QoE and engagement, the aim of this work is to bridge those two fields together, preparing the way to combine their advantages in new models.

2.2 QoE Model for Video Streaming

While an initial delay can be observed in any environment, stalling, which has the highest impact on the QoE, only occurs in networks with very little bandwidth. Since we want to focus on these two parameters, we must consider a low bandwidth scenario in the following. Our model does not include quality adaptations since these can only occur in networks with sufficient bandwidth. There exists a QoE model that describes the impact of stalling on the QoE [50] and a separate model that describes the impact of the initial delay [33]. In this section, we bring both models together using two different approaches to create a combined QoE model. This is required to assess the impact of different buffer settings.

2.2.1 Stalling QoE Model

The QoE of HTTP streaming depends mostly on the number of stalling events N and average length L of stalling events. A QoE model combining both key influence factors into a single equation $f(L, N)$ is provided in [50] and found to follow the IQX hypothesis [60] describing an exponential relationship between the influence factors and QoE. In particular, the model function returns mean opinion scores (MOS) on an absolute rating scale of 5 points, with 1 indicating the lowest QoE and 5 the highest QoE.

$$f(L, N) = 3.5e^{-(0.15L+0.19)N} + 1.50 \tag{2.1}$$

Due to well-known rating scale effects, the model in Equation 2.1 has a lower bound of 1.50, as users avoid the extremities of the scale called “saturation effect” as is discussed in [87]. In contrast, if the video is not stalling, no degradation is observed and users rate the impact of stalling as ’imperceptible’, i.e., a value of 5.

It must be noted that the model function in Equation 2.1 is based on subjective user studies with videos of duration up to $T = 30\,\mathrm{s}$. For other video durations, the normalized number $N^* = {}^{N}/_{T}$ of stalling events has to be considered which

requires to adapt the parameters $\alpha = 0.15$ and $\beta = 0.19$ in Equation 2.1, respectively.

As the goal of our investigation is the analysis of the impact of different user profiles, we parameterize the function in Equation 2.1 with α and β and conduct a parameter study on their impact. For the sake of simplicity, we normalize the QoE value to be in the range $[0; 1]$. As a result, we arrive at Equation 2.2 as parameterized QoE model Q_1 to quantify the impact of stalling on QoE for different user profiles expressed by α and β. Thereby, the parameter α adjusts the sensitivity of the user to the total stalling duration $L \cdot N^*$, while β quantifies the sensitivity of the user to the frequency of stalling events.

$$Q_1(L, N^*) = e^{-(\alpha L+\beta)N^*} = e^{-\alpha L N^* - \beta N^*} \tag{2.2}$$

The model function Q_1 in Equation 2.2 has the same form as Equation 2.1 and follows the IQX hypothesis but allows to investigate different user profiles. For example, some users may suffer stronger from interruptions which is then adjusted by a higher value of β. Thus, a user profile is expressed by α and β.

2.2.2 Initial Delay QoE Model

Another impairment on HTTP streaming QoE are initial delays before the video playout starts. The impact of initial delays T_0 is modeled by the following function g and the model parameters are obtained from subjective tests [33].

$$g(T_0) = -0.963 \log10(T_0 + 5.381) + 5 \tag{2.3}$$

The results in [33] show that the impact of the initial delay is independent of the video duration which was either 30 s or 60 s in the user tests. Further, it was observed that users have a clear preference of initial delays instead of stalling and that service interruptions must be avoided in any case, even at costs of increased initial delays for filling up the video buffers.

For the sake of simplicity, we normalize the function in Equation 2.3 yielding to the QoE model Q_2 for initial delays T_0, such that Q_2 returns values in $[0; 1]$ and that $Q_2(0) = 1$. The user profile is parametrized with γ determining the impact of initial delays. The constant $c = 5.381$ is taken from 2.3 defining the shape of the curve. Since the logarithm is not bounded, only positive values are considered to ensure $Q_2(T_0) \in [0; 1]$.

$$Q_2(T_0) = -\gamma \log10 (T_0 + c) + \gamma \log10 (c) + 1 \tag{2.4}$$

2.2.3 Combined QoE Model

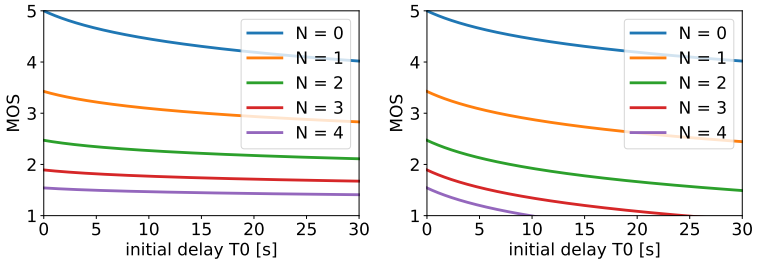
For dimensioning the video buffers, we are interested in a simple, parameterizable QoE model which considers both, the impairments due to stalling and due to initial delays of the video playout. Therefore, we suggest the following model Q. Since the impact of stalling events clearly dominates the user perception [33, 50], we consider the following rationale for the combined QoE model. A user facing an initial delay T_0 experiences a QoE value of $Q_2(T_0)$. If additional stalling events occur, this will lower the QoE further. Thus, $Q_2(T_0)$ is the upper bound of QoE. For N^* stalling events with an average length L, the QoE will be further decreased by $Q_1(L, N^*)$.

There are now two evident possibilities for realizing the decrease, 1) by multiplying the values Q_2 and Q_1, or 2) by subtracting the values. It is an open issue in the QoE research community whether to combine multiple QoE degradations in an additive or multiplicative way [88].

Published in [59], we find the combined multiplicative QoE model Q_* based on the rationale above yields to Equation 2.5 which again returns values in $[0; 1]$.

$$Q_*(T_0, L, N^*) = Q_1(L, N^*) \cdot Q_2(T_0) \tag{2.5}$$

Figure 2.2(a) depicts exemplarily QoE[1] depending on the number N^* of stalling events and the initial delay T_0 for an average stalling event of $L = 2\,\mathrm{s}$. As user profile, we use $\alpha = 0.19$, $\beta = 0.15$, and $\gamma = 0.3$. The blue line in Figure 2.2(a) shows the impact of the initial delay T_0 on the QoE when no stalling occurs. When compared to the the other lines which include stalling, it can be seen that stalling dominates the impact on QoE, as the differences between the curves with different T_0 diminish for more stalling events.



(a) Multiplicative model Q_* from Equation 2.5.

(b) Additive model Q_+ from Equation 2.6.

Figure 2.2: QoE models combined from Q_1 and Q_2 for $L = 2\,\mathrm{s}$ and the user profile $\alpha = 0.19$, $\beta = 0.15$, and $\gamma = 0.3$.

ITU-T[2] Recommendation P.1201 proposes an additive QoE model for non-adaptive HTTP streaming which is referred to as buffer-related perceptual indicator in the Appendix III [89]. This model follows the same rationale above, start from the maximum QoE value which is $1 = Q(0, 0, 0)$, subtract the degradation $1 - Q_2(T_0)$ stemming from initial delay, and from stalling $1 - Q_1(L, N^*)$.

[1]For the sake of readability, we present the QoE values in all figures on the common 5-point MOS scale by linearly mapping the model values to $[1; 5]$.

[2]International Telecommunication Union Telecommunication Standardization Sector

Then, we arrive at the following additive QoE model Q used in the analysis [4, 59].

$$\begin{aligned} Q_+(T_0, L, N^*) &= 1 - (1 - Q_1(L, N^*)) - (1 - Q_2(T_0)) \\ &= Q_1(L, N^*) + Q_2(T_0) - 1 \end{aligned} \tag{2.6}$$

Figure 2.2(b) shows the results for the same parameters as used above, cf. Figure 2.2(a). It can be seen that the initial delays have now a more severe impact than for the multiplicative model. Further, the additive model does not follow the IQX hypothesis anymore. As observed in subjective studies [60, 90] the subjective sensibility of the QoE is the more sensitive, the higher this experienced quality is. If the QoE is very high, a small disruption will decrease strongly the QoE. If the QoE is already low, a further disturbance is not perceived significantly. The additive model does not respect this relationship anymore. Nevertheless, we will analyze the video buffer dimensioning based on the multiplicative and the additive QoE model, as this allows to differentiate the importance of initial delays on QoE. However, as we will see from the results in the next section, both QoE models lead to the same conclusions.

For the purpose of simplification, we assume that the impact of α, β on the QoE and the impact of initial delay are independent of each other. The impact of the initial delay is according to [33] if no stalling occurs.

2.3 User Preferences and Usage Scenarios

In this section, we try to answer the following research questions:

- Do we need to know the QoE preferences of the user in order to optimize QoE? E.g., whether a user prefers few long stalling events over many short stalling events.

- Do we need to know the user's behavior or the usage scenario in order to optimally dimension the video buffer? E.g., video browsing or whether a user only watches a few seconds of a video.

We answer these questions by creating an $M/M/1$ queuing model that simulates an HTTP video streaming player to estimate stalling and initial delay. We use this model to conduct a mean value analysis for three scenarios regarding different user parameters and three watching scenarios and map the results to MOS values by applying the multiplicative QoE model from Section 2.2.

2.3.1 System model

We provide a system model for video playback, in order to study the stalling behavior of HTTP video streaming. We consider the playback of a video consisting of multiple frames. The frames are downloaded in-order and arrive at the client with rate λ while the playback time is given by the video framerate μ, resulting in an offered load of $a = \frac{\lambda}{\mu}$. Here, a quantifies the available network bandwidth normalized by the video framerate.

In order to reduce the number of stalling events during playback, the video player uses a playback buffer. Video playback stops, if less than q frames are currently available for playback and is only resumed if the buffer contains $p = q + d$ frames. The normalized buffer size d^* (in video seconds) relates the buffer size d (in frames) to the video framerate μ, i.e. $d^* = \frac{d}{\mu}$.

Next, we introduce metrics used to evaluate the influence of the playback buffer parameter selection. The relative amount of time spent in stalling compared with the total duration of the playback process including stalling is given by the stalling ratio R and the number of stalling events normalized by the video length N^*. For the case of finite videos, we furthermore consider the stalling duration L which gives the sum of times spent in stalling states during the complete video playback.

To derive the key performance metrics, we model the system as a $M/M/1/\infty$ queueing model with *pq*-policy in Section 2.3.1. This Markov model allows to

derive the following application layer metrics for video streaming that are relevant to the QoE. The player is assumed to always be in one of two states: playing or stalling. The average length of a stalling period is given as L. The average length of a playing period is given as the busy period B. The relative amount of time spent in stalling compared with the relative amount spent replaying the video is given as the ratio of buffering events R. The number of stalling events normalized by the video duration is given as the normalized buffering ratio N^*. An overview of the notation is given in Table 3.4. A mean value analysis allows us to investigate the impact of system parameters in the steady state and also in the transient phase for the analysis of short (finite) videos and user aborts.

M/M/1 Queue With pq-Policy

The state of the video playback is characterized by the tuple (i, j), where $i \in \{0, 1\}$ is the playback state of the client, i.e. the video is not played back if i is 0 and the video is played back if i is 1 and $j \geq 0$ gives the number of unplayed frames currently available at the client. Furthermore, we give the probability of the playback being in state (i, j) as $x(i, j)$. For the general case $0 \leq q < p$ We obtain the following equilibrium state equations.

$$
\begin{aligned}
\lambda x(0,0) &= 0 \\
\lambda x(0,i) &= \lambda x(0,i-1) && i \in [1,q) \\
\lambda x(0,q) &= \lambda x(0,q-1) + \mu x(1,q+1) \\
\lambda x(0,i) &= \lambda x(0,i-1) && i \in (q,p) \\
(\lambda+\mu) x(1,q+1) &= \mu x(1,q+2) \\
(\lambda+\mu) x(1,i) &= \lambda x(1,i-1) + \mu x(1,i+1) && i \in (q+1,p) \\
(\lambda+\mu) x(1,p) &= \lambda(x(0,p-1) + x(1,p-1)) \\
&\quad + \mu x(1,p+1) \\
(\lambda+\mu) x(1,i) &= \lambda x(1,i-1) + \mu x(1,i+1) && i \in (p,+\infty)
\end{aligned}
$$

Table 2.1: Notion and variables frequently used.

Video and network input parameter	
B	network data rate (kbps)
V	video encoding bitrate (kbps)
λ	network bandwidth normalized by mean frame size, i.e. network frame arrival rate
μ	video bitrate normalized by mean frame size, i.e. video frame rate, with $\mu > \lambda$
a	offered load $a = \lambda/\mu$ captures the relative network bandwidth (normalized by video bitrate)
Player parameter and status variables	
X	number of frames in video buffer
Y	status of video player; $Y = 0$ means stalling, $Y = 1$ means playing
$X(Y, i)$	probability that the number of frames in video buffer is i and the status of the player is Y
p	video buffer threshold when video playout starts (frames), i.e. $X \geq p \Rightarrow Y = 1$
q	video buffer threshold when video playout stalls (frames), i.e. $X \leq q \Rightarrow Y = 0$; it is $0 \leq q < p$
d	buffer size $d = p - q$
d^*	buffer size normalized by video bit rate $d^* = d/\mu$
K	maximum queue size of video buffer (frames)
Key performance parameters	
F	stalling frequency (1/s)
N	number of stalling events
T	video duration
N^*	number of stalling events normalized by video duration
R	stalling ratio
L	stalling length (s)
Q	QoE value

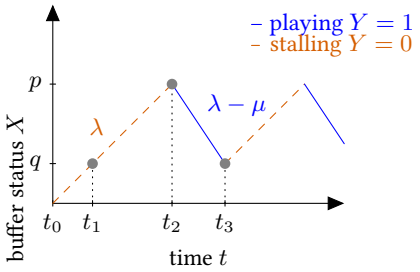


Figure 2.3: Mean value analysis of video buffer status evolving over time.

The state probabilities can be obtained analogously to [91].

$$
\begin{aligned}
x(0,i) &= 0 && i \in [0,q) \\
x(0,i) &= \frac{1-a}{d} && i \in [q,p) \\
x(1,i) &= \frac{a(1-a^{i-q})}{d} && i \in (q,p] \\
x(1,i) &= \frac{a^{j-p+1}(1-a^d)}{d} && i \in (p,+\infty]
\end{aligned}
$$

From this we obtain the stalling ratio R as the probability of being in a stalling state, i.e.

$$R = \sum_{i=0}^{p-1} x(0,i) = 1 - a\,. \tag{2.7}$$

Mean Value Analysis of Steady State

A mean value analysis of the $M/M/1/\infty$ queueing model with pq-policy is now conducted which can be derived in order to obtain the duration of stalling events L and the number of stalling events N^*.

In Figure 2.3 we see that the initial download begins at t_0 and new frames arrive with rate λ at the client. The number of frames in the buffer exceeds q the first time at t_1. At time t_2, the rebuffering goal p is reached for the first time and playback begins. While the download of new frames continues with rate λ, frames are played out with rate μ, resulting in a buffer change with rate $\lambda - \mu$. Thus, the number of buffered frames reaches q at time t_3. This process repeats which results in an alternating chain of stalling and playback phases.

In this analysis we consider the steady state, i.e. especially neglecting the time $t_1 - t_0$. First, we consider the time required for the buffer to fill from q frames to p frames, i.e. obtaining d frames while no playback is occurring. This time depicts the average duration L of a single stalling event. This is given as the time between t_1 and t_2, and we get

$$L = t_2 - t_1 = \frac{p-q}{\lambda} = \frac{d}{\lambda} = \frac{d^*}{a}\,. \tag{2.8}$$

The average stalling length L only depends on the actual buffer size d and the network bandwidth λ. Next, we consider the time required for the number of frames in the buffer to decrease from p to q, i.e. the time between t_2 and t_3, $t_3 - t_2 = \frac{d}{\mu-\lambda}$. Combining these two equations we get the time between two stalling events as $t_3 - t_1 = (t_3 - t_2) + (t_2 - t_1) = \frac{\mu d}{(\mu-\lambda)\lambda}$.

The stalling ratio R follows as

$$R = \frac{t_2 - t_1}{t_3 - t_1} = 1 - a\,, \tag{2.9}$$

yielding the same result as in 2.7 in Section 2.3.1.

Finally, we can obtain the number of stalling events normalized by video duration by analyzing the busy periods of the system. Here, the mean idle period is given by $L = \frac{d}{\lambda}$.

For the mean busy period B it holds $\frac{B}{B+L} = 1 - R = a$, which yields $B = \frac{a}{1-a}\frac{\lambda}{d}$ and the normalized number of stalls,

$$N^* = \frac{1}{B} = \frac{\mu - \lambda}{d} = \frac{1-a}{d^*}\,. \tag{2.10}$$

Equation 2.10 can also be derived by considering $N^* = \frac{1}{t_3 - t_2}$. While N^* relates the stalls to the video duration, the stalling frequency F denotes the number of stalls per time. It holds $F = \frac{1}{t_3 - t_1} = aN^*$ which is also equal to $F = x(0, p-1)\lambda$ to change the player with the state probability $x(0, p-1)$ and the network arrival rate λ. However, from an end user's point of view, the metric N^* but not F is of importance.

Beside the network bandwidth λ[3] and the video bitrate μ, the number N^* of stalling events depends only on the video buffer size $d = p - q$, but not on the concrete values of p and q in the steady state.

2.3.2 QoE Study for Typical User Scenarios

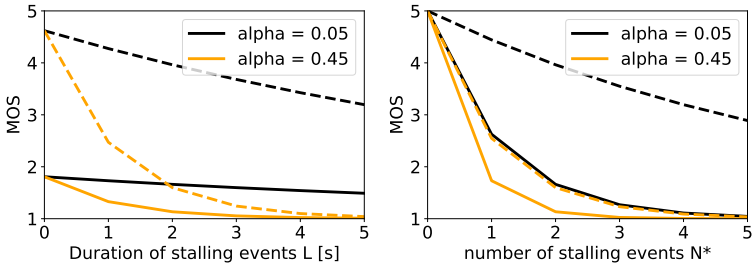
Typical usage scenarios of video streaming services reflect the following user behaviors:

- *Watch Later*,
- *Regular Video Streaming*,
- *Video Browsing*.

For these scenarios, the video buffer size d^* is optimized concerning QoE and the impact of the user profile (α, β) is analyzed for values close to those obtained in a user study [50] ($\alpha = 0.15, \beta = 0.2$) that represent average users. For the following parameter study, we additionally use values that are arbitrarily deviated to indicate users with different preferences ($\alpha \in \{0.05, 0.45\}, \beta \in \{0.05, 0.8\}$).

[3] For the sake of readability, we use the term 'network bandwidth λ' instead of 'network bandwidth in terms of frames' or 'network frame arrival rate'.

The influence of these parameters on the QoE can be seen in Figure 2.4. The difference between the steady state (Section 2.3.1) and the finite case (Section 2.3.1) is 0.2 points on a 5-point MOS scale for 30 s videos. For the *Watch Later* and *Regular Video Streaming* scenario, we assume longer video durations and can use the steady state results. In contrast, for *Video Browsing* short viewing times of 10 s require the finite case results. For the sake of readability, we transformed the QoE value linearly to be in the range $[0; 1]$.



(a) Impact of L on the MOS ($N^* = 2$).

(b) Impact of N^* on the MOS ($L = 2s$).

Figure 2.4: Influence of user parameters α and β on the MOS based on Equation 2.2. The solid line represents $\beta = 0.8$, the dashed line $\beta = 0.05$.

Watch later Scenario

In the 'watch later' scenario, a user requests a video, but the user does not expect that the video playout starts immediately. This may be the case for example when the user wants to watch an HD movie even though the network bandwidth is low. During that initial delay, the user may do something else, e.g. opening another web page in a parallel tab in the browser or getting some snacks in the kitchen. Thus, QoE is not affected by initial delays and we only need to consider

Q_1 in Equation 2.2. In the steady state, it is $L = {}^{d}/_{\lambda}$ and $N^* = \frac{\mu-\lambda}{d}$ and we obtain the following QoE relation in Equation 2.11.

$$Q_1(L, N^*) = e^{(\mu-\lambda)(\alpha/\lambda+\beta/d)} = e^{-\alpha\frac{1-a}{a}-\beta\frac{1-a}{d^*}} \tag{2.11}$$

Since the QoE function in Equation 2.11 is strictly monotonically increasing with the buffer size d^*, the optimum is achieved for $Q_+ = \lim_{d^*\to\infty} Q_1(L, N^*) = e^{-\alpha\frac{1-a}{a}}$. Thus, the QoE value only depends on the parameter α in the limit. To see for which buffer size we are close to the optimum, we consider the relative difference $\frac{Q_+-Q_1(L,N^*)}{Q_+}$ when it is less than $\Omega = 5\,\%$. This is true for $d^* > -\beta\frac{1-a}{\log(1-\Omega)}$.

For $\beta \in \{0.05, 0.2\}$, a small buffer size of $d^* > 4\,\mathrm{s}$ is already enough to be close to the optimum Q_+ for any offered network condition a. For users extremely sensitive to stalling ($\beta = 0.8$) buffer sizes up to 15 s are required. However, a buffer of 4 s is sufficient for a relative difference to the optimum of 20 %. In general, the larger the buffer size the better the QoE is in this scenario. In practice, a buffer size of 4 s is a good choice.

Default Video Streaming Scenario

In the case of normal streaming, the user wants to watch a video immediately. In contrast to the 'watch later' scenario, the initial delay now impacts the QoE according to Equation 2.6. Figure 2.5 shows QoE depending on the buffer size for the streaming scenario and different user profiles in a network situation $a = 0.5$ leading to a stalling ratio $R = 0.5$. Now, QoE optima exist for finite buffer size, if the impact of the initial delay is taken into consideration. We notice that α does increase the QoE but has no significant impact on the optimal buffer size. In contrast, for different β we observe different optima for the buffer size. Therefore, we can ignore α when optimizing the buffer size regarding the QoE. A buffer size less than 0.5 s results in a severe loss of QoE for all users. A buffer size of 2-4 s offers a good QoE for the average user and any sensitive user. Increasing

the buffer size further decreases the QoE. In practice, QoE is only marginally improved if the buffer is adapted to the user's specific preference (see resulting optimal QoE values for different β values in Figure 2.5).

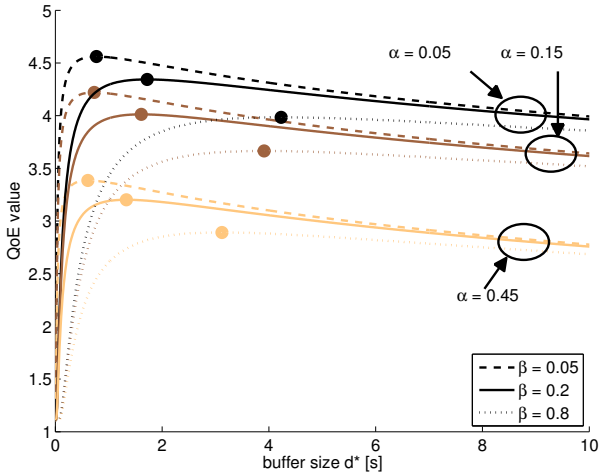


Figure 2.5: Dimensioning of buffer size in the Streaming Scenario for available network bandwidth of $a = 0.5$. Maxima marked as dots mainly depend on β.

Video Browsing Scenario

In the case of video browsing, the user watches a video for a short period of time. This includes cases such as, viewing a short video completely, viewing a short part of a long video or skipping ahead in a video frequently (thus watching multiple short parts of a video). Since we know from the previous section that α and β have a marginal impact on the optimal QoE, we consider only the default parameters $\alpha = 0.15$ and $\beta = 0.2$ in the following. However, for video browsing, the impact of the initial delay may be more important for the user. Therefore,

we consider $\gamma = 0.3$ corresponding to Equation 2.3 as well as a delay sensitive user $\gamma = 0.6$.

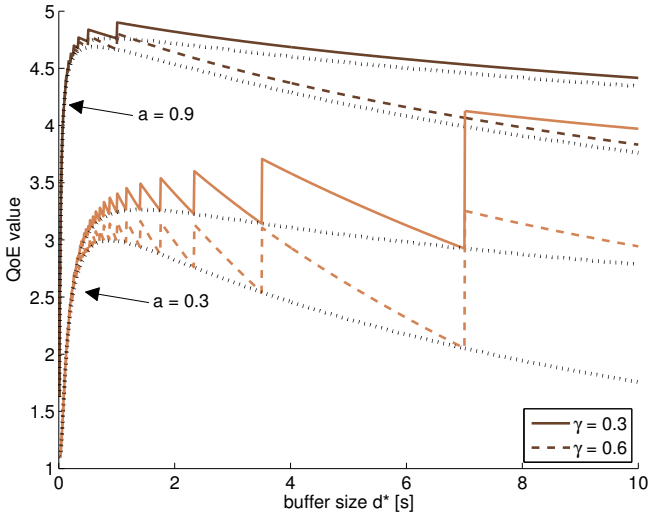


Figure 2.6: Dimensioning of buffers for Video Browsing users with varying QoE sensitivity to initial delays ($\gamma = 0.3, 0.6$) in two network situations ($a = 0.9, 0.3$). Users abort the video after 10 s. Steady state results i.e. for long videos are indicated by dotted lines

In Figure 2.6, the impact of the buffer size on the QoE is depicted for the case that the video is aborted after the first 10 s. Multiple local QoE maxima exist independently of γ, which appear when the number of stalls changes. The results for the steady state are also included. We observe that the steady state represents a worst-case buffer dimensioning, but there is little difference between steady state and the finite case. However, for larger buffer sizes, the difference between the local maxima and the steady state increases. Nevertheless, in those cases, the

initial delay exceeds tens of seconds. So, this scenario cannot be described as realistic video browsing.

In general, if the exact viewing length of a video was known (e.g. short videos will be watched completely), the buffer size could be set so that the QoE lies at a local maximum which is independent of γ. However, this method can result in a severe loss of QoE (depending on γ) if the user aborts earlier. In practice, a buffer size of 1-2 s is recommended for video browsing. If the buffer size is set too large, γ determines again the actual QoE loss.

2.4 Practical Evaluation of QoE

In this section, we conduct a measurement study in which we evaluated the playback behavior of a HAS video player in real-world scenarios. The goal of this study is to investigate the impact of buffer configurations on the QoE in a practical, realistic setting, where adaptation may occur. The Shaka player[4] allows us to configure the buffer configuration[5] and the adaptation heuristic[6], this way we can compare different buffer configurations. For the purpose of this study we left the adaptation heuristic at its default setting. The bandwidth conditions, that the players face, are mobile scenarios, which were recorded in everyday commuting situations, using different means of transport. To investigate the impact on the end user, we map the found application layer QoS measures to a MOS using our QoE model from Section 2.2. Furthermore, we introduce a maximum buffer size, whereas previously the buffer was considered to be unlimited. This has the advantage of less wasted traffic if a user abandons the video early.

[4] https://github.com/google/shaka-player
[5] https://shaka-player-demo.appspot.com/docs/api/tutorial-network-and-buffering-config.html
[6] https://shaka-player-demo.appspot.com/docs/api/shaka.extern.html#.AbrConfiguration

2.4.1 Methodology

We developed a setup that allows us to test HTML video players under predefined bandwidth conditions, while monitoring their behavior. In these experiments, we used the Shaka player v.2.0.1 for HAS. In the following, we describe the elements of our setup.

The client can be an arbitrary device running a recent web browser that is able to run the video player. In our experiments this is a *Firefox 50* browser on a machine running *Ubuntu 16.04*. It could also be changed to any other device, for example a smart phone, to investigate whether the players behave differently on other device types.

The experiment is controlled by the 'Control server', which provides the web page for the client with the video player and collects the results. In case of the Shaka player, the video is hosted using a local video server. With the YouTube player, we had to use YouTube as video source. Because of this, we had two differently encoded versions of the same video. This is further described in Section 2.4.1.

The bandwidth is controlled by bandwidth traces, which reflect real-world scenarios. These were provided by Riiser et al. using a notebook and a 3G modem for measuring download speed, and a GPS module for localization [92]. The team investigated bandwidth rates on different mobile travel situations. During the measurements, files were downloaded via HTTP constantly. As this is the same technique used by adaptive video streaming the same behavior would be experienced by a video streaming client. The bandwidth traces reflect situations as they are typical for commuting, using different ways of transportation (Bus, car, ferry, metro, tram, and train). We focus on mobile scenarios, because in these the bandwidth is varying, while in stationary situations, there is often more bandwidth available, which also has a lower variation over time. Numbers about the bandwidth scenarios can be found in Table 2.2. These bandwidth traces are

reproduced using *tc*[7] which configures the packet scheduler of the Linux kernel of a dedicated Linux machine.

Table 2.2: Mean and standard deviation of the bandwidth traces.

scenario	mean	standard deviation
Bus	244.7 kB/s	164.1 kB/s
Car	67.3 kB/s	90.8 kB/s
Ferry	161.0 kB/s	146.2 kB/s
Metro	72.1 kB/s	74.5 kB/s
Train	134.2 kB/s	119.4 kB/s
Tram	100.4 kB/s	62.4 kB/s

The video players use different video sources. While the video source of the Shaka player can be configured arbitrarily, the YouTube player can only play videos from YouTube. As a result, we had to use different test videos. The video for the Shaka player was provided by a local web server. The YouTube player fetched a video directly from YouTube. Because this relies on an Internet connection, during the experiments the Internet connectivity was continuously monitored to detect impairments.

Both video players used the same test movie with a length of 9 min 54 s, but in a different encoding and different representation layers. While the video for the Shaka player comes with three quality levels (Average data rate 79.99 kB/s, 178.60 kB/s, 482.69 kB/s), the YouTube video has seven quality levels (Average data rate 13.41 kB/s, 30.17 kB/s, 40.70 kB/s, 80.08 kB/s, 150.60 kB/s, and 268.72 kB/s). This means the YouTube player has a lower minimum and maximum bandwidth and more possibilities to adapt the video quality to the current available bandwidth.

The mode of operation is as follows: the bandwidth traces of one scenario are infinitely repeated by the bandwidth limiter. At the same time, the video playback is repeated continuously, so that it starts at different positions of the

[7] http://manpages.ubuntu.com/manpages/xenial/en/man8/tc.8.html

bandwidth traces. Using the players' API, every second the current playback status, the buffer level, and video quality are recorded, which are then sent to the control server after a video playback has finished. Additionally, this is also done whenever a stalling or adaptation event occurs. This way, we obtain a precise history of the players' behavior.

2.4.2 Measurement Results

The Shaka player allows to configure the buffer, so that we could compare its behavior. A buffer configuration consists of two settings: the maximum buffer size, which determines the maximum length of video in the buffer, and the buffer size p, which is the minimum amount of video in the buffer required to resume playback after a stalling event, e.g., in Section 2.3.1 the maximum buffer size was infinite and for the buffer size values of $d^* \in]0\,\mathrm{s}\,10\,\mathrm{s}]$ were analyzed. As all experiments are run in real-time, we limited the number of buffer configurations to four:

- Maximum buffer size 10 s, buffer size of 2 s, which is the default configuration of the Shaka player
- Maximum buffer size $90\,\mathrm{s}$, buffer size of $10\,\mathrm{s}$
- Maximum buffer size $180\,\mathrm{s}$, buffer size of $10\,\mathrm{s}$
- Maximum buffer size $180\,\mathrm{s}$, buffer size of $20\,\mathrm{s}$

In Figure 2.7, the number of adaptation events per bandwidth and buffer configuration is plotted. An adaptation event means that the player changes the playback quality thus selects another representation layer. The number of adaptation events clearly differ between the different bandwidth traces. The bandwidth scenarios with the highest numbers of adaptation events are 'Bus', 'Ferry', and 'Train' with a median number of adaptation events between five and six, respectively three and four, and two in case of 'Train'. Those bandwidth scenarios are the three with the highest average bandwidth, and play a significant share of the video in medium quality, while in all other scenarios nearly all the

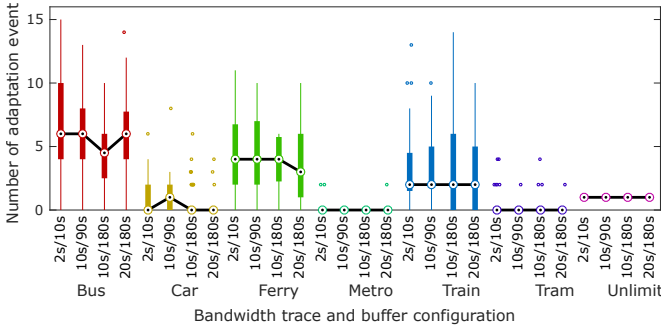


Figure 2.7: Shaka player: adaptation events

time the lowest quality is played. As it can be expected, with unlimited bandwidth, always one single adaptation event occurs, when the player switches from the initial quality to the highest representation layer. It is notable, that the buffer configuration has no influence on the number of adaptation events.

Figure 2.8 shows the distribution of the resolutions, in which the video was played. In case of unlimited bandwidth, most of the time the video was played at maximum quality. When the bandwidth was limited, the highest quality was never played, most time the lowest quality was selected. Only in the scenarios 'Bus', 'Ferry', and 'Train' the medium quality was played for a significant time. It can be assumed, that only in these scenarios the bandwidth was high enough, see Table 2.2. In scenarios with low average bandwidth, like 'tram' and 'metro', only the lowest quality was played, while in case of unlimited bandwidth, after an initial quality switch, only the highest quality was played. It can be stated, that the buffer configuration does not influence the played quality. Therefore, the resolution and the number of quality switches are not considered any further, the QoE evaluation focuses on the stalling events and the initial waiting time.

Next, we take a look at the stalling events. Figure 2.9(a) shows their number, while Figure 2.9(b) shows their duration. It can be seen, that in all cases, where the

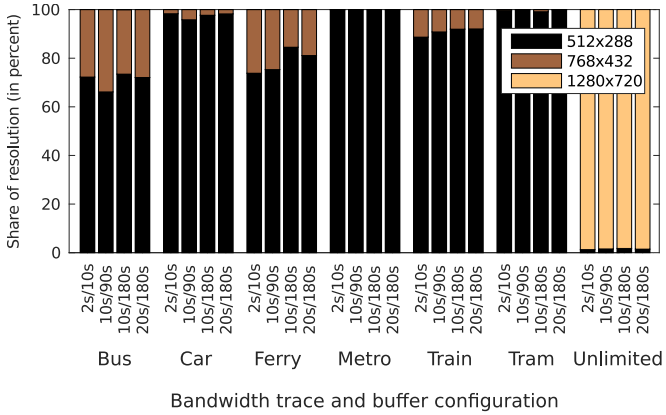
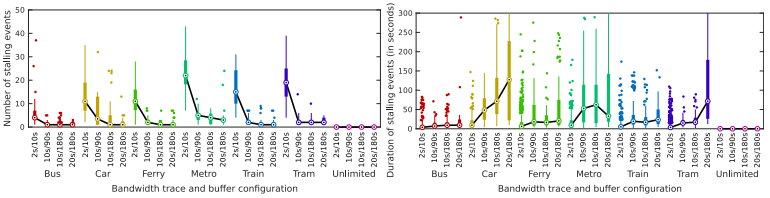


Figure 2.8: Shaka player: played resolution

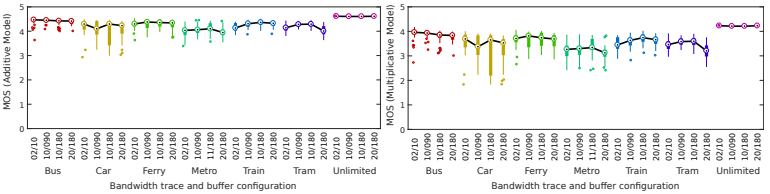
bandwidth was limited, the lower the buffer size was set, the more stalling events occurred. Especially in the configuration with the smallest buffer, in median there were many of stalling events. Compared with the lowest configuration, there is a huge improvement using the maximum buffer size of ten seconds and the 90 s buffer size. Whereas, there is no significant reduction of the number of stalling events between a buffer size of ten and 20 seconds and a maximum buffer size of 180 s. In the case of unlimited bandwidth, no stalling occurred. When looking at the duration of the stalling events, it can be found that the median length of the stalling events was significantly longer with a higher buffer size. The exception in the 'Metro' scenario and the 20 second buffer size can be explained by the methodology, where the video starts at arbitrary positions of the bandwidth traces. In this case, the variance is much higher than the cases with a smaller buffer and a lower buffer size.

For an overall evaluation, where both the initial waiting and the stalling are considered, these two models have to be combined. To do so, there are different methods, which were presented in Section 2.2, namely the additive and



(a) number of stalling events (b) duration of stalling events

Figure 2.9: Shaka Player: stalling events.



(a) additive model (b) multiplicative model

Figure 2.10: Shaka Player: combined QoE using the additive and the multiplicative model.

the multiplicative model. In Figure 2.10(a) the MOS of the different scenarios and buffer configuration ratings are combined using the additive model. The multiplicative model is used in Figure 2.10(b). For simplicity, we weigh initial waiting and stalling the same. In practice, this has to be adapted to the use case. If users tend to browse more, skip large parts in the videos and abandon the playback, the initial waiting time is more important, while when users select a video and watch it for a longer time, the effect of stalling becomes more important.

In most bandwidth scenarios, the buffer configuration with a 10 s buffer size and a maximum buffer size of 180 s has the best MOS rating. Only in the 'bus' scenario, which has the highest average bandwidth, the configuration with the

smallest maximum buffer size (10 s) has a slightly better rating, because in this specific scenario the initial waiting time lowers the high rating of the stalling events, because in this scenario only little stalling occurs. This evaluation using the QoE also matches the results of previous evaluations in [13], where the playback statistics were manually interpreted. This shows, that the QoE model can be used in practice for evaluation of adaptive video streaming. The automatic interpretation and reduction of complex statistics to a single MOS value allows it to include QoE evaluation and monitoring into practical scenarios.

2.5 Buffer Policies

This section aims to study the relationship between User Engagement and the QoE in on-demand video streaming and how these metrics are affected by different buffer policies. We conduct this investigation via a queuing model that describes the video player behavior in terms of stalling periods for arbitrary network conditions and video characteristics. The results are mapped to QoE according to our QoE model. Further, we propose a model for User Engagement that is established by fitting existing measurement results. Based on this model, we analyze the correlation between QoE and User Engagement numerically. In addition, we compare three video buffer policies analytically and in a trace-driven simulation.

2.5.1 User Engagement Model

User Engagements describes the activity or attention of users in a system. As described in Section 4.1.1, for video streaming we restrict the definition of the User Engagement to the average amount of time U (in minutes) users watch a video, including stalling events. In [25], several data sets were collected and analyzed. In Figure 2.11 we take a closer look at one of their data sets: LvodA which contains long VoD clips with a length of about 35 min to 60 min. In each data point users with the same ratio of buffering events are related to an average

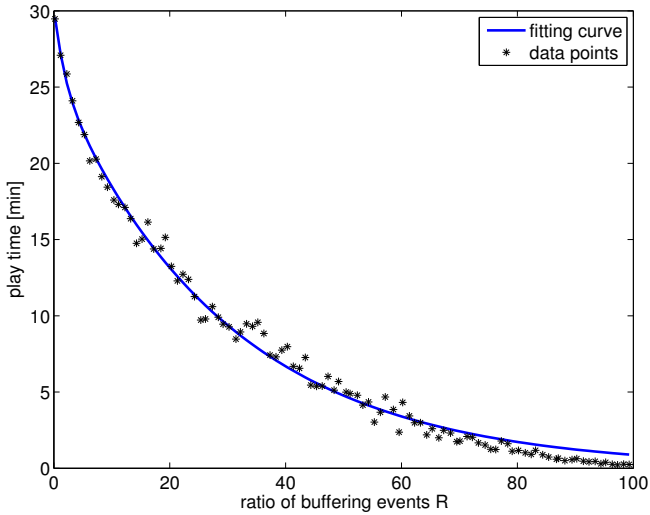


Figure 2.11: Fitting based on the buffering ratio R. Data taken from [25].

play time. We fitted a nonlinear curve to these data points in least-squares sense using MATLAB, which provides us with a fitting function

$$U(R) = 4.2712 \cdot e^{-0.5435 \cdot R} + 25.9000 \cdot e^{-0.0339 \cdot R} \tag{2.12}$$

This function maps the ratio of buffering events R to the average play time in minutes. For the fitting, we chose a double exponential decay since this is commonly used for describing spontaneous human behavior (e.g. in [93]). The Pearson correlation coefficient for this fit is 0.996 (Spearman 0.997). The RMSE is $0.659\,\mathrm{min}$ (normalized RMSE $0.092\,\mathrm{min}$). This indicates that the fit is very accurate.

2.5.2 Impact of Buffer Policies on QoS and QoE

In this section we present an analytical approach to calculating key QoS parameters for the player model described in 2.3.1. A detailed mean value analysis of the steady state was derived in Section 2.3.1 for an $M/M/1$ model leading to the following QoE model

$$Q(L, N^*) = 3.5 \cdot e^{-(\alpha \frac{1-a}{a} + \beta \frac{1-a}{d^*})}.$$

In the following, we extend this model to an $M/G/1$ model for three buffer policies: the D-policy, the n-policy, and the T-policy. In [94], the authors derive the distributions and the means of the busy and idle periods of queuing models for these three policies. In the following, we adapt these results for HTTP video streaming for the case of a generally distributed service process. For the sake of clarity, when variables for specific policies are discussed, they have the policy name as their index, e.g., for the D-policy, we use the notation L_D, B_D, N_D^* instead of L, B, N^*.

M/G/1 with D-policy

With a D-policy, the idle period ends if the sum of the service times of the units in the queue amounts to D. For the specific case of video streaming, this policy means that stalling ends after the data in the buffer amounts to a certain play time D. This policy guarantees that the length of the busy period is at least D. For the D-policy, it is

$$E[L_D] = \frac{1}{\lambda}(M(D) + 1)$$

with $M(D)$ being the renewal process of D,

$$E[B_D] = \frac{M(D) + 1}{\mu - \lambda},$$

$$E[N_D^*] = \frac{1}{B_D} = \frac{\mu - \lambda}{M(D) + 1}.$$

It follows

$$Q(L_D, N_D^*) = 3.5 \cdot e^{-(\lambda-\mu)(\frac{\alpha}{\lambda} + \frac{\beta}{M(D)+1})} + 1.5.$$

If we assume the buffer size d does not change during a video session, then $M(D) = d - 1$ is constant as well. Thus $L = L_D$ and $N^* = N_D^*$ are equal for $M/M/1$ and $M/G/1$ with D-policy. Therefore, $Q(L, N^*) = Q(L_D, N_D^*)$ is equal for both models.

M/G/1 with n-policy

With the n-policy, the idle period ends if $n = d^* \cdot \mu$ bytes are in the queue. For the n-policy, it is

$$E[L_n] = \frac{n}{\lambda} = \frac{d^*}{a},$$

$$E[B_n] = n \cdot \frac{1}{\mu(1-a)} = \frac{d^*}{1-a},$$

$$E[N_n^*] = \frac{1}{B_n} = \frac{1-a}{d^*}.$$

Since $E[L_n]$ and $E[N_n]$ are equal for M/M/1 and M/G/1 with n-policy, $Q(L, N^*) = Q(L_n, N_n^*)$ is also equal for both models.

M/G/1 with T-policy

With the T-policy, when an idle period starts, a timer is started. If the timer reaches T, the systems verify whether a unit arrived during the idle period. If it did arrive, the busy period is started. Otherwise, the timer is restarted. The probability that no unit arrives during the idle period T is $e^{-\lambda T}$. If we can assume $e^{-\lambda T} = 0$, this policy guarantees that the length of each stalling period is exactly T. For the T-policy, it is

$$E[L_T] = \frac{T}{1 - e^{-\lambda T}},$$

$$E[B_T] = \frac{\lambda T}{(1 - e^{-\lambda T})(\mu - \lambda)},$$

$$E[N_T^*] = \frac{1}{B_T} = \frac{(1 - e^{-\lambda T})(\mu - \lambda)}{\lambda T}.$$

It follows

$$Q(L_T, N_T^*) = 3.5 \cdot e^{(1-\frac{1}{\alpha})(\alpha+\beta\frac{1-e^{-\lambda T}}{T})} + 1.5.$$

If we can assume $e^{-\lambda T} = 0$ and choose $T = \frac{d^*}{a}$ this leads to $Q(L_T, N_T^*) = Q(L, N^*)$. In Figure 2.12 we see that the impact of the policies is small for $T = d^*$. Consequently, the service process has no impact on the QoE under the assumption of a Markovian arrival process. This means that only the mean and not the variance of the video bit rate matters for the QoE, assuming an $M/G/1$-model. This result was also observed in simulation results that will be presented in Section 4.3.2.

2.5.3 Analytic Results

This section takes a closer look at the relation between QoE and User Engagement by discussing analytic results for the queueing model described in 2.3.1. In addition, we look at the simulation of the download of a real video in a real network and compare it with the analytic results.

First, we focus on the D-policy as it reflects current video player implementations of HTTP streaming, and investigate the impact of reception rate (i.e. offered load) and different buffer sizes on QoE and User Engagement. Later, in Section 2.5.4, we compare the different policies in terms of QoE and User Engagement.

Figure 2.13 shows how the offered load (or ratio between network bit rate and video bitrate) a is related to the MOS value and the play time for different buffer sizes d^* (e.g. a value $a = 0.5$ means that the bandwidth is half of the video bit rate). We notice that increasing the offered load a leads to an increasing average

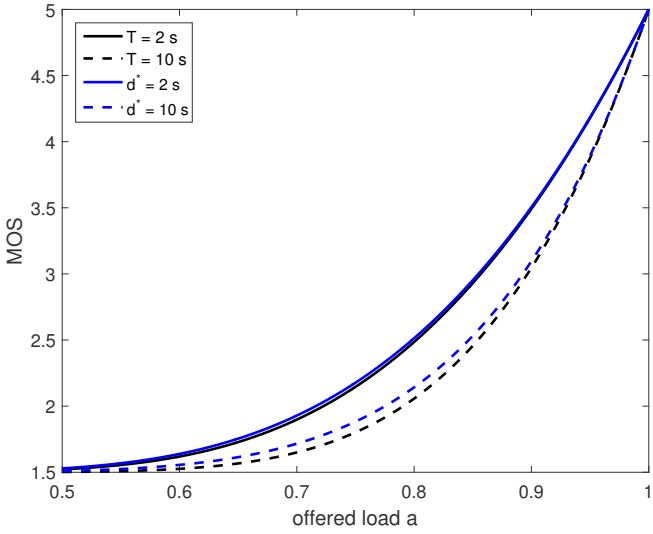


Figure 2.12: Difference between T-policy and D-policy if $T = d^*$ is chosen. The n-policy is omitted for the sake of clarity.

MOS and User Engagement. It should be noted that MOS values lower than 2.5 are not considered acceptable by most users [95]. In addition, we see that an increasing buffer size leads to higher MOS with the optimum being reached at $Q_+ = \lim_{d^* \to \infty} Q(L, N^*) = e^{-\alpha \frac{1-a}{a}}$ as shown in [4]. In contrast, the buffer size does not have an impact on the User Engagement. A large difference between QoE and User Engagement is that for $a < 0.4$ the MOS is 1.5 and does not change while increase the play time is noticeable. This is because the QoE model that we use is based on short video clips while the user engagement model is based on long videos. User Engagement has been observed to be lower for shorter videos [74].

Next, we investigate how the QoE value is related to the User Engagement. In Figure 2.14 we calculated the User Engagement and the QoE for various

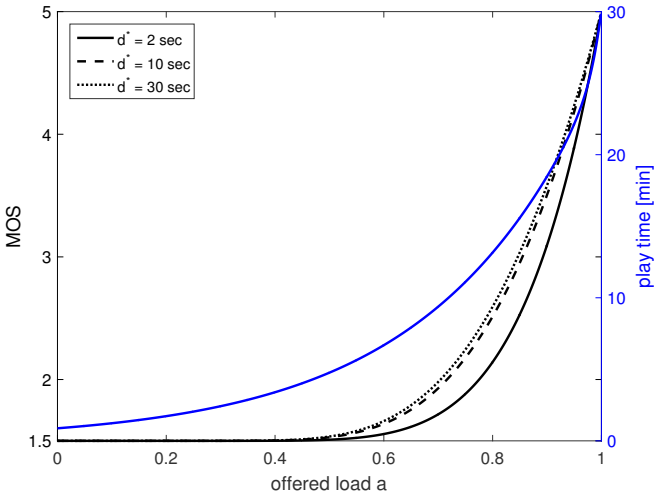


Figure 2.13: QoE value in an $M/G/1$ system. The offered load quantifies the ratio between average network bandwidth and video bit rate.

offered loads. We observe that an increase in QoE always leads to an increase in User Engagement. Since our model for User Engagement does not take the buffer into account, more research is necessary in order to identify its impact. Furthermore, we notice that for very low QoE values, it is difficult to estimate the User Engagement as users may react differently in such scenarios.

The Pearson correlation coefficient is 0.981 (Spearman 0.994). A larger buffer leads to a higher mean play time for the same QoE. This can be explained by the fact that an increase in buffer size leads to an increase in QoE, but not to an increase in mean play time. This means that users will abort videos much earlier if the QoE is low. Therefore, it is critical to ensure a high QoE if User Engagement is to be maximized.

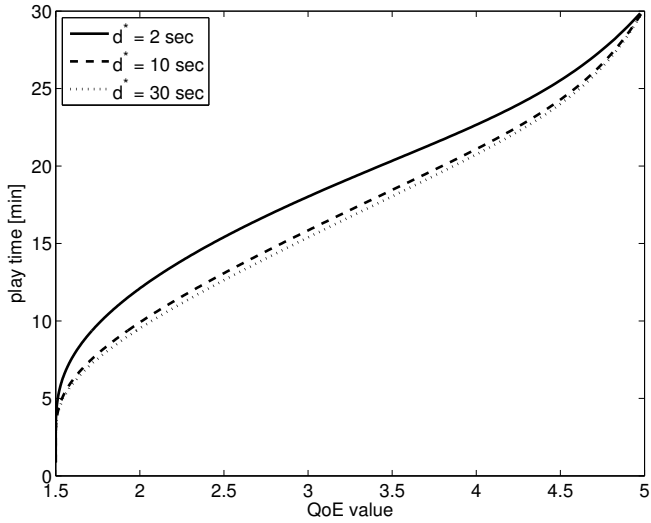


Figure 2.14: User Engagement in relation to the QoE in an $M/G/1$ system for $d = 2, 10, 30$ according to Equation 2.12 in Section 2.5.1.

2.5.4 Simulation Results

In order to compare our analytic results with measurement results, we simulated replaying a real video using a real network trace that was recorded in [96]. For this simulation we chose the video "Tears of Steel" in a low spatial resolution (320×180). It is a $12\,\mathrm{min}$ short movie with a variable bit rate. The network trace was recorded by downloading a large file via HTTP using a UMTS stick while driving on a highway. The resulting trace has a strongly fluctuating bit rate. In total, we used 30 different traffic patterns that were created by adding a temporal shift to the original traffic pattern in [97]. We simulated different network capacities by adjusting the video bit rate, resulting in various offered loads a.

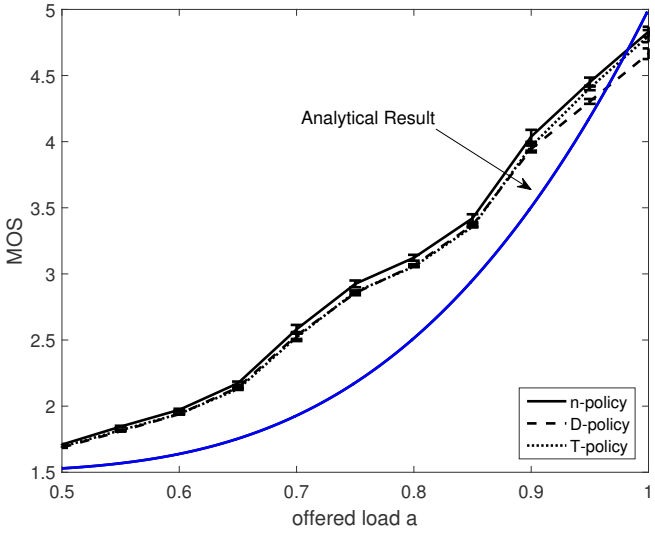


Figure 2.15: Relation between MOS and offered load a for three buffer policies. Simulation results for real video and real traffic pattern (black) and analytical results for $M/G/1$-model (blue). Policies have little impact on MOS.

In our simulation, video frames are downloaded with a rate that is based on the effective network capacity and the size of the video frame in a best effort manner. Video frames are replayed at a constant rate of 24 frames per second until the video ends. If a stalling event occurs, it is resolved according to the given buffer policy. The simulator is implemented in MATLAB and is available online[8].

In the following, we compare the simulation results to the analytic results. Figure 2.15 shows the impact of the offered load a on the MOS for the n-policy, the D-policy and the T-policy. It is clearly visible that the policy does not impact the QoE value significantly. In addition, the real traces lead to a higher QoE

[8] https://github.com/ChristianMoldovan/HAS-Simulator

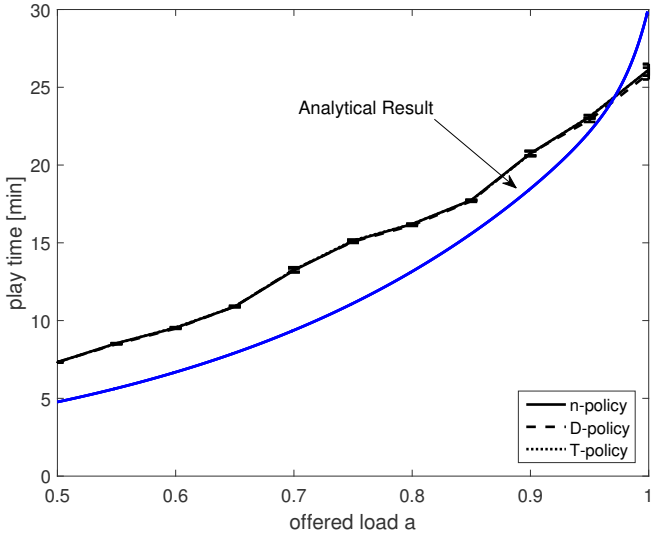


Figure 2.16: Relation between play time and offered load a for the three buffer policies. Simulation results for real video and real traffic pattern (black) and analytical results for $M/G/1$-model (blue). Policies have no impact on play time.

than the $M/G/1$ model. This is mainly due to video specific attributes, i.e. the distribution of frame sizes. Nevertheless, we consider a generally independent distribution for the frame sizes in videos. While more advanced models may lead to more realistic results, they cannot be solved analytically.

In Figure 2.16 we investigate how the buffer policies impact the User Engagement that was calculated based on the rate of buffering events R according to Equation 2.12 in Section 2.5.1. The main observation is that the policies have almost no impact on the mean play time. This means that since the T-policy does not require any information from the player, it provides a solid alternative to the

other policies. This is particularly the case if hiding such information becomes a common practice in the future.

2.6 Lessons Learned

In this chapter, we investigated how different buffer dimensions impact the application layer QoS, the QoE and the User ngagement.

The rebuffering goal determines the degree to which the buffer must be filled before playback begins at the start of the video or after a stalling event. Thus, this parameter impacts the initial delay, the length of stalling events and the duration of playback events. To describe its impact on the QoE, we describe how a unified QoE model for the initial delay and stalling can be created. To evaluate the impact of the rebuffering goal, we define an $M/M/1$ model and conduct a parameter study where we investigate three typical video streaming scenarios. We modeled user preferences with parameters in the QoE model and found that they only have a minor impact on the optimal buffer settings. They can thus be ignored for practical purposes. However, the viewing scenario should be considered. In particular, it has to be differentiated if the user is video browsing (recommended buffer size 1-2 s) or watching video immediately or later (4 s).

We then conduct a measurement study where we use vehicular traces to evaluate the real-world behavior of different buffer configurations. For this, we use the shaka player since it allows modifications to the buffer. In this study, we investigate the impact of the buffer size and the maximum buffer size on application layer QoS and QoE. We see that increasing the buffer above a certain size does not increase the played video quality. In contrast, more data can be wasted if the buffer is large, in particular if the available bandwidth is very high. Therefore, we recommend to strictly limit the buffer size if the bandwidth is much higher than the video bit rate.

In order to identify how the User Engagement is impacted by such factors, we find a good fit between the buffering ratio and the time spent watching videos. Next, we investigate three policies that can control when the video is resumed

after a stalling event. For this we extend the $M/M/1$ model for the video player to an $M/G/1$ model and prove that the QoE and User Engagement is equal in both models for all buffer policies. In a simulation we compare these results for the steady state with results for a 12 min movie that is watched with a network trace recorded in a mobile scenario. We find that the differences in the results of the Markov model and the simulation are small, so that the $M/G/1$ model describes the real behavior sufficiently well. Furthermore, we show that there is a strong correlation between the QoE and User Engagement.

3 Optimizing Adaptation in Video Streaming

A current challenge of video service providers is the delivery of content in high quality while avoiding stalling events. It is well known that stalling events and the video encoding bit rate, i.e. the video resolution, have a significant impact on the acceptance rate and the QoE [98]. Therefore, it is important for the service provider to develop a sophisticated adaptation logic which can prevent stalling events even when faced with congested links during after-work hours or unstable Internet connections, such as cellular access. Even though HAS can avoid stalling in most cases, a high number of quality switches may also have a negative impact on the QoE [40, 99]. Very frequent adaptation events lead to an ever-changing image quality in the video, which is tough on the user's eye. Therefore, we want to keep the number of quality changes or switches as low as possible while providing high playback quality. A very passive adaptation strategy plays the video on a lower quality in order to keep the number of switches at a minimum. While some users may prefer watching a video with a high average quality with many switches over low quality with few switches, there is little research on the trade-off between quality and switches.

A great deal of research has focused on the QoE-driven design of adaptation algorithms, with the goal to deliver video content at high quality levels while avoiding stalling, long initial delays, and frequent quality switches. Faced with the question of how to benchmark the performance of HAS adaptation algorithms compared to a theoretical QoE optimum, the authors in [97] propose problem formulations to compute the theoretical optimum for both single- and multi-

user scenarios. Their addressed multi-user scenario assumes users concurrently watching and downloading the same video over a shared bottleneck link.

In this chapter, the overall research questions that we tackle can be formulated as follows: *To what degree can the QoS be optimized in HAS?* An overview of the QoS parameters that we investigate and the methods that we apply to reach this goal is given in Figure 3.1. As a first step, we want to find out how the number of quality levels impacts the QoS of HAS in a network with high bandwidth variation. For this purpose, we present a queueing model for a generic buffer based adaptive bit rate scheme that we use to determine the state probabilities from which we derive key performance indicators. Next, we want to determine how much room for optimization there is with current adaptation algorithms. Thus, we compare an adaptation algorithm which was deployed by YouTube with a heuristic and two optimal approaches in terms of QoS. Furthermore, we analyze the trade-off between average video quality and quality switches with regard to a weighting parameter. For this purpose, we use a quadratic program, that optimizes adaptation in video streaming towards higher quality and fewer quality switches. This is done with respect for the user's individual preference for these two QoE impact factors. We conduct a user-centric analysis of fairness for several simultaneous HAS clients. We answer the following questions: (1) To what extent are stalling and the video quality affected when video streaming clients share a network bottleneck? (2) Is the simultaneous playback of multiple adaptive video players fair in terms of QoE? To answer these questions, we conduct a measurement study where multiple users, who share a bottleneck, are watching videos at the same time over a long period. We focus on QUIC, since it is used in Chromium / Chrome which is the most spread web browser. In this study, we observe situations where clients are treated unfairly. Our next goal is to provide the means for benchmarking such solutions in the context of multiple users accessing Video on Demand content while sharing a bottleneck link. For that purpose, we propose a quadratic problem formulation to compute the theoretical optimum in terms of adaptation strategies and corresponding segment downloads across multiple users under given bandwidth constraints. By

aiming to maximize both service quality and fairness, we quantify and compare the impact of different fairness objectives (bandwidth fairness, pattern fairness, and session fairness) on resulting quality and achieved QoE fairness. Based on conducted simulations and parameter studies, our results demonstrate the benefits of optimizing for session fairness as compared with other approaches. Furthermore, we propose three linear programs that use three approaches for viewport prediction in 360° videos. The three approaches consist of statistics-based viewport prediction, linear extrapolation of the current user's viewport, and a neural network that learns typical head movement patterns for specific videos by training with other user's viewports. Based on which approach is used, different information is needed from the network, the user and the video service provider. The proposed application-aware and network-aware linear programs optimize the video resolution and the number of resolution changes.

The remainder of this chapter is based on content that has been published in [8, 3, 9, 10, 14, 20] and is structured as follows. First, we discuss background and related work on adaptation strategies. Next, we propose a generic queueing model for buffer based adaptive streaming. In the next section, we investigate by how much YouTube could improve its quality optimally using an optimization problem. In Section 3.4, we extend this problem to analyze the trade-off between quality and switches. We then turn towards the aspect of fairness in adaptive streaming and investigate the performance of the QUIC protocol in a scenario with multiple concurrent YouTube players that share a bottleneck. In Section 3.5, we propose an optimal solution for such a scenario under various fairness aspects. Section 3.6 discusses how different approaches to viewport prediction can be used for the optimization of 360° video streaming. Finally, we discuss lessons learned in Section 3.7.

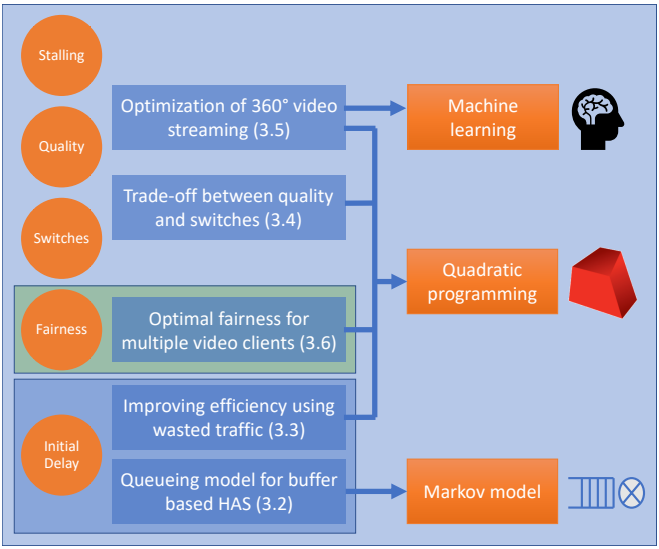


Figure 3.1: Contributions of this chapter by with QoS parameters and methodologies.

3.1 Background and Related Work

3.1.1 Modeling Video Streaming

There are many different approaches that are suitable to conduct a performance evaluation in video streaming. An overview of a selection of important references is given in Table 3.1.[62] uses two models that assume steady-state for Netflix videos. In their first model, they assume constant segment durations, while their second, idealized model assumes infinitesimal segment durations, so that bit rate adaptation can be applied continuously. [61] uses an epoch-based model for multiple players where each player chooses a new bit rate at the start of a new epoch. A similar scenario is investigated in [100], where a Markov model is presented that is used to evaluate sharing policies for network-assisted HTTP adaptive streaming. [101] uses a Markov model to investigate the influence of user behavior in cellular networks on the buffer starvation in video streaming. Burger et al. [102] models the video buffer as a $GI/GI/1$ queue with pq-policy using discrete time analysis. They use this model to study stochastic properties of the buffer level distribution and to evaluate the impact of network and video bitrate dynamics on the video QoE. Another approach is to determine the optimal adaptation strategy using a linear program. Such programs are used in [103] for the optimal rate allocation for video streaming in wireless networks with user dynamics. In [3], a linear program uses context information to avoid stalling in a tunnel scenario.

Control theory is another approach that works well with live streaming [104] and in cases of varying bandwidth [105]. In [106], the authors use a control theory model for adaptive video streaming to assess the optimally reachable QoE. Furthermore, they propose an adaptation algorithm that is based on throughput as well as the buffer state. The authors of [107] investigate Akamai's adaptive streaming system and model the playout buffer as a control system. The paper [108] is an early work that uses control theory to model the video buffer as a single server queue with constant service rate. The performance of adaptive video streaming can also be described with flow-level models [109]. These approaches

usually aim at improving the QoE or key performance indicators of QoE such as the impact of the chunk duration in mobile networks.

Table 3.1: Models of HTTP Adaptive Streaming

reference	goal / scenario	approach
[62]	video streaming	user study
[61]	multiple user scenario	epoch based model
[100]	cooperative policies for streaming	Markov model
[101]	influence of user behavior on buffer	Markov model
[110][111][109]	impact of network on app. QoS	flow level model
[112]	heterogenous traffic and stability	flow level model
[113]	chunk duration	flow level model
[105]	QoE under time-varying bandwidth	control theory
[106]	throughput-based adaptation	control theory
[104]	optimal QoE in live streaming	control theory
[107]	Akamai's adaptive streaming	control theory
[108]	model for video buffer	control theory
[103]	multi-user rate allocation	linear program
[114]	quality, switches	linear program
[97]	quality, switches, multi-user	linear program

In [97, 115], Hoßfeld et al. conclude that avoiding stallings is the first priority when optimizing a HAS service for user experience. The second and third priority are the average video quality shown to the user and minimizing the number of switches and the amplitude of the switches. In [116], Nam et al. conduct a large scale study on YouTube and confirm the high (harmful) impact of re-bufferings and quality switches on the user's QoE. Further related work in the area of HAS QoE and on HAS in general can be found in [56].

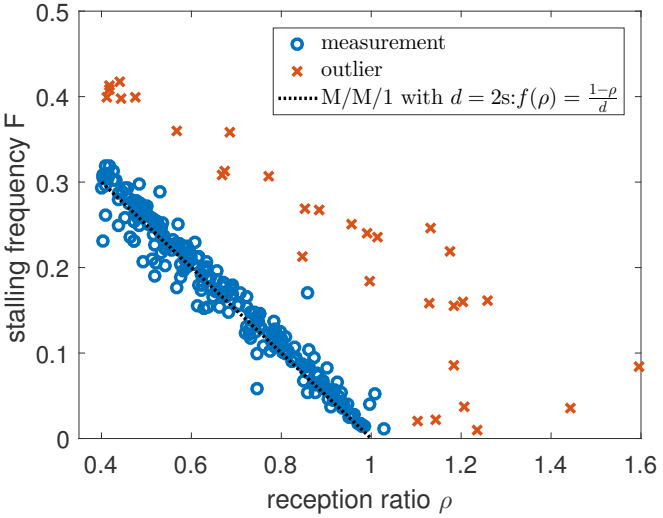


Figure 3.2: Measurement study of YouTube stalling characteristics [55] compared with an analytical M/M/1 model [4]. The variable d describes the amount of video content that must be downloaded before the video resumes playback after a stalling event.

In [98], Casas et al. conclude that the ratio between video bit rate and downlink bandwidth significantly influences YouTube's adaptation. They show that YouTube's adaptation is not robust in bottle-necked scenarios. Yao et al. show in [117] that the iOS YouTube player uses overlapping segments to smoothen the playback. Rao et al. [118] and Ito et al. [119] evaluate YouTube's traffic pattern during video playback. They show a dependency of the behavior on the viewing device. In [120], Añorga et al. show that YouTube uses a large playout buffer of 13 s to 40 s and therefore can only adapt slowly to changing bandwidth conditions. In [121], Alcock et al. describe YouTube's initial burst phase in detail. They show that 32 s of playback time in a low quality level is transferred to the client as fast as possible before the transfer is throttled. We account this for a major

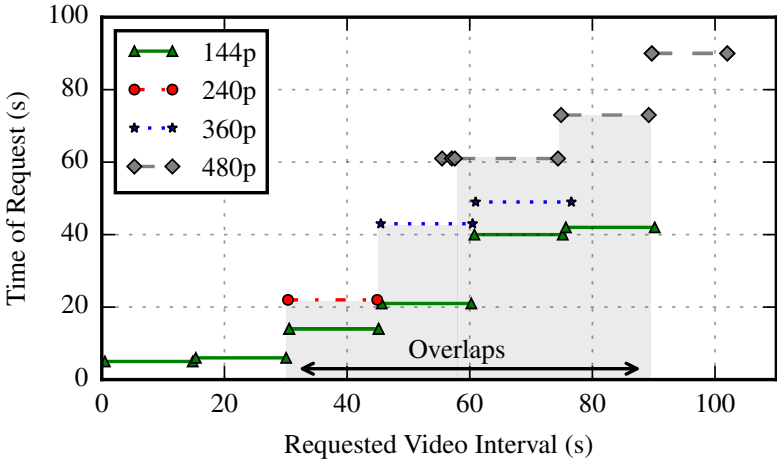


Figure 3.3: Example request schedule from one of the experiment runs [125]. From 30 s to 90 s overlaps can be observed where low quality (144p) is replaced by higher quality levels (240p, 360p and 480p).

source of redundant traffic as the low quality level is replaced later by higher quality segments. In [122], Mansy et al. evaluate YouTube's adaptation behavior in terms of redundant traffic, playback behavior and bandwidth utilization. In a wireless scenario with one video and one bandwidth pattern they quantify the redundant traffic to 16%. They also show that the adaptation strategies of other content providers behave in a similar way. Lui et al. [123] conclude that YouTube's buffer level on mobile devices is based on the amount of data buffered, not on the amount of playback seconds. They observe redundant traffic when segments at the beginning of the video are re-downloaded and quantify the redundancy to 15%. In [124], the authors identify additional redundant traffic on the transport layer of YouTube in a mobile scenario. They quantify the redundant traffic to 35% due to frequent termination of TCP connections and in-flight packets.

Figure 3.3 shows a request schedule for one of the experiment runs. The x-axis shows the request video interval in playback time, e.g. the first 16 s of the video. The y-axis shows the time of request based on the experiment time, with 0 being the time the first HTTP GET request was sent to the server. At first, the player requests one minute of the lowest quality level. Then, 20 seconds into the experiment, the player revises its previously made decision, discards two of the low quality segments (i.e. 30 s of playback time) and starts to download a higher quality level instead. The shaded areas in the figure illustrate where lower quality segments were discarded. The figure illustrates that in this video view out of 105 s of video, approximately 60 s were available in more than one quality level at the player. As is shown in a previous study [125], this is not an isolated incident but happens on a regular basis.

3.1.2 Optimization of HAS

HAS can also be formulated as an optimization problem, e.g., with linear or quadratic programming. The first quadratic program that solved adaptive streaming was presented in [114]. In a first step the optimal quality is determined that is reachable without stalling. In a second step, the number of quality switches is minimized while playing at the optimal quality and while avoiding stalling. This program was extended in [97] to include multiple viewers who watch the same video at the same time. In this chapter we present many extensions to these programs that also take into account fairness and context or consider new scenarios such as 360° video streaming.

3.1.3 Fair Video Streaming

In a system where scarce resource must be shared, fairness is always a concern. While each player has its own requirements and internal status like throughput and buffer level, it only has a limited view of the whole situation. It is not aware of neither the overall network situation nor other players, but only knows its own status, in particular the video qualities it is able to select, the buffer level, and the

speed with it downloads further segments. So, every player makes its own choice depending on its own view of the situation. As a result, one player may choose to play a high quality, while other players decide to switch to a lower quality, resulting in an unfairness between players or stalling events as the available bandwidth is overestimated. An enhancement for adaptive video streaming called 'SAND' was standardized [126], which allows direct communication between server and the video player and thus allows arrangements between these parties. In general solutions that use TCP (such as [127]) perform very well in terms of fairness, while non-UDP based protocols (such as QUIC) have difficulties with multiple players in highly varying networks. An adaptation algorithm presented in [128] tries to eliminate these scalability issues by creating coalitions between players that participate in a collaborative game.

This problem of multiple video players sharing a bottleneck link is enhanced by the mode of operation of HAS and the buffers that are used. For example, two players are playing different videos simultaneously. Now one player finishes playing, so the other player can use the full bandwidth, increase the video quality, and fill its buffer. If the first video player at the end of the video now starts to play another video, it has a serious disadvantage: first, the link is already saturated, it has no knowledge about the bandwidth of the link, and its buffer is empty, so that it cannot compensate bandwidth fluctuations. This issue was investigated previously in [129], where in practical experiments the behavior of a simple adaptive video streaming player was investigated. There, it was already found that multiple video players influence each other, resulting in instability of the playback. As a main problem the on-off behavior of video players was identified, as HAS players can only estimate the available bandwidth during download. Six different HAS clients are compared in [130] in a HAS experiment testbed. The study focuses on the adaptation logic, the throughput estimation, and the fairness in a scenario with multiple players. Their evaluation shows that YouTube behaves selfish in a competition with Netflix and Vimeo players and obtains more bandwidth due to its UDP-based transport. Our work differs from this work by focusing on challenging mobile scenarios with high throughput

variations. In contrast to [130], we rely on real network traces. Furthermore, we have a deeper investigation of the fairness, where we find a significant difference between players during runs.

Mitigations for this problem can be either taken on server-side, like proposed in [131] or in the network, for example with OpenFlow [132]. But both approaches enhance the complexity of the video streaming setup. One advantage of HAS is that the adaptation logic works on client-side, so that the server can be completely stateless, so that is well-suited for Content Delivery Networks (CDNs). So, the appropriate would be to modify the adaptation logic accordingly, like done in [61].

In a measurement study, [133] compare QUIC and TCP when transferring data over the same bottleneck link at the same time. In their study, they use various emulated network conditions, desktop and mobile clients, and multiple historical versions of QUIC. They use web browsing and YouTube video streaming as test applications. Similar to us, they use a very long video. They find that QUIC consumes almost twice the bandwidth than all other TCP flows combined. Furthermore, they claim that two clients that use QUIC are fair to each other within the same system which differs from the results for YouTube video streaming. However, they do not study multiple QUIC flows in detail. The authors of [134] also investigate the behavior of QUIC and TCP with multiple users that share the same bottleneck link. They investigate the QoE performance and the QoE fairness of three DASH algorithms within a testbed environment. In addition to their testbed, they conduct experiments in the public Internet with background traffic. For their experiments they use a 5 min video. Similar to our study, their clients start with a small time offset. Their results show that a system with multiple TCP DASH clients is more resilient than a system with QUIC DASH clients when competing for traffic.

Jain's fairness index is a popular fairness index for QoS [135] that determines the ratio between the square of the mean and the mean of the squares for a set of values. However, it is only suitable for measures on a ratio scales [43], i.e. an interval scale with a true zero point. Resources are allocated according to

max-min fairness among users if a user only receives more resources when no other user will suffer and obtain less [43]. A rather novel fairness index for QoE has been presented in Hoßfeld et al. [136]. Hoßfeld's fairness index is defined via the ratio between the observed standard deviation of QoE values and the maximal possible standard deviation $F = 1 - \sigma/\sigma_{max}$ which makes it suitable for any interval scales. In Section 4.3.2 we use this index to quantify fairness from a user-centric point of view for quality layers that serve as QoE indicators. A detailed discussion of QoS and QoE fairness is provided in [43] focusing on the notion of fairness in shared environments and in networking, as well as fairness from the user's perspective.

3.1.4 360-Degree Video Streaming

With the advent of VR, 360° videos have become more popular during the last years. A 360° video, also called omnidirectional video or spherical video, is a video where every angle is recorded at the same time using an omnidirectional camera. In VR applications, such videos are typically viewed on an HMD. For example, they can be applied to increase immersion in panoramic videos or for the rehabilitation of cognitive and motor abilities [137]. Although 360° videos currently make out a small ratio of total videos, VR is one of the most trending Internet applications, such that 360° video streaming traffic will significantly increase.

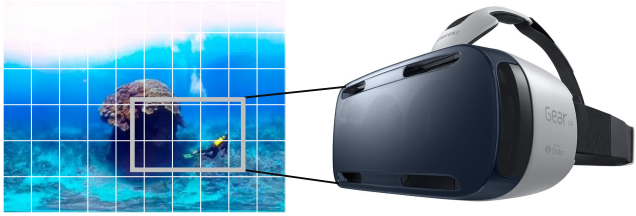


Figure 3.4: Viewport of an HMD on an equirectangularly formatted 360° video. The viewer only sees a subset of all tiles at any point in time.

Current video platforms always stream the whole video in the same visual quality, although in the case of 360° videos only a small share of the total video, the so-called viewport, is watched at any point in time by the user, compare Figure 3.4. To reduce the data consumption of 360° video streaming, approaches have been developed to predict which viewport the user will watch. If the viewport can be predicted accurately, it is sufficient to stream it in high visual quality while the remaining tiles can be transmitted in a lower quality, as they are less likely to be watched. However, an accurate viewport prediction is only possible if the video consists of short video segments [138], which leads to high segment overhead. A segment length of 2 seconds is suggested in [139] as a good trade-off between segment overhead and head movement prediction accuracy. A description of the basic principles of adaptive tile-based streaming of omnidirectional video services over HTTP, available encoding options, and evaluations with respect to bit rate overhead, bandwidth requirements, and quality aspects can be found in [140]. To predict the performance of 360° videos in terms of application layer QoS and QoE the authors of [141] propose PERCEIVE, a method which uses machine learning techniques.

Several works relied on viewport prediction in the context of 360° video streaming. The authors of [142] developed a framework for 360° videos that uses 35% less data while providing similar quality in slow viewport movement scenarios. First viewport-adaptive streaming algorithms were presented in [143] and [144]. The authors of [145] used a trajectory-based approach which groups past users that have a similar viewing trajectory and create a model of the viewport evolution over time for the identified groups. This model is used at prediction time for new users. The authors of [146] proposed a heatmap-based model and a trajectory-based model for viewport prediction. Each user received a weight that determines the importance of his viewports for the final contribution. The authors showed that using other user's information increases the performance over only using the current user's information. A similar trajectory-based prediction scheme and an adaptation algorithm for VR videos were presented in [147]. In [148], it was suggested to investigate head movement prediction extracted from

the feedback of previous users. A 360° head movement data set from 59 users was provided in [149]. The authors of [150] used a probabilistic model of viewport prediction to reduce side effects caused by wrong head movement prediction. Their approach significantly reduced stalling with small buffers. In addition, they provided an optimization problem that minimizes quality distortions and spatial quality variance. A first approach to using neural networks for viewport prediction was presented in [151]. They leveraged content- and sensor-related features for the prediction. To verify their approach, they conducted a user study with 25 viewers who watched ten 360° videos. Their approach consumes less bandwidth and has a shorter initial delay than other approaches. In [152], a viewport prediction model was presented that is based on a convolutional neural network (CNN). In addition, they presented a trajectory prediction model which is based on an RNN. The RNN was combined with a correlation filter-based viewport tracker that adds content awareness to increase the performance of the prediction. An approach that relies on learning contextual bandits to predict the viewport for 360° videos is presented in [153]. While other approaches follow the behavior of the current user, the authors of [154] believe that user behavior is hardly predictable and is mainly correlated to moving objects in the video. With this idea in mind, they develop a motion tracking algorithm that identifies representative moving objects. The authors of [155] separated video segments into two tiers: the basic whole video tier and the viewport tier. The whole video tier can be downloaded early and can be stored in the buffer while the viewport tier is downloaded on demand to enhance the current viewport.

3.2 Queueing Model for Buffer-Based HAS

In this chapter, we first present a novel queuing model for buffer-based video streaming adaptation that allows us to determine the state probabilities of the buffer and to derive key performance indicators. We chose to model this problem with a queueing model since it allows us to model the dynamic characteristics of network traffic. The state of the player is modeled with a two-dimensional Markov

chain for variable segment durations. The states of this queue are defined by the number of segments that are in the buffer and the layer of the video segment that is currently being downloaded. Using this model, we answer the question how the number of quality layers impacts the frequency of stalling events.

The arrival rate or download rate of the segments depends on the size of the segments and the bandwidth of the client. It determines with which rate the buffer is filled. If segments are replayed faster than they are downloaded, then the buffer is reduced. In a video player that does not employ adaptive mechanisms the buffer is depleted and the video stalls. In adaptive videos streaming, the video will request segment that are downloaded in a lower quality which leads to much fewer stalling events. Since segments are then downloaded in a lower bit rate, segments arrive at a higher rate. For our model, we assume that segment arrivals follow a Markovian arrival process.

The length (i.e. duration) of video segments determines how often adaptation can be done. Netflix segments have a constant duration of 4 seconds [62] and YouTube segments have constant duration of about 5 seconds. On the other hand, it is possible to use variable length segments for adaptation instead of fixed length. For example, it could be possible to use I-frames for adaptation. This would be possible, as long as an I-frame exists at the same position in both quality representations.

We use four throughput settings from $0.5\,\mathrm{Mbit/s}$ to $5\,\mathrm{Mbit/s}$. For comparison, one can look at the Netflix ISP Speed Index for US as of February 2020. The Netflix ISP Speed Index is the average bit rate of Netflix content for 'Prime Time' with cellular networks being exempted from the measurements[1]. The highest average bit rate that was measured was $4.73\,\mathrm{Mbit/s}$ and the lowest $3.49\,\mathrm{Mbit/s}$.

In HTTP adaptive streaming, videos are available in a given number or layers with different bit rates. However, not all available layers may be selected by the adaptation strategy. E.g., a 4k resolution can only be requested by selecting it manually on YouTube. In the data set that we use for the YouTube study provided in [156], all videos were requested in at least 5 quality levels which is why we

[1] https://ispspeedindex.netflix.com/about/

limit the investigation to these five levels. YouTube uses 2–6 quality levels [52] while Netflix uses 12–15.

The switching threshold of a buffer based adaptation strategy determines at which buffer thresholds different quality levels are requested. Thresholds that are close to each other lead to frequent quality switches which impacts the QoE negatively. A study of current video streaming platforms and adaptation algorithms is conducted in [52]. On YouTube the initial bit rate/quality depends on the size of the player window, while on Netflix it depends on the country. In a user study, it was found, that the QoE can be improved by selecting an appropriate initial bit rate [52]. For our model, we assume that it always starts on the lowest layer. In contrast, all of our analysis assumes steady state, so the initial layer has no impact on our analytical results.

3.2.1 Model Assumptions and Limitations

Sieber et al. [156] observed that YouTube requests multiple segments at once and later may re-download segments in a higher resolution if there is enough bandwidth available. While this leads to redundant traffic, the viewing experience is greatly increased in a high bandwidth scenario. We model a specific buffer based strategy, which does not allow to download segments more than once. In addition, our model represents a purely buffer based algorithm. For the steady state, the buffer is the only parameter that needs to be observed according to [62]. Furthermore, our proposed model always starts on the lowest layer and does not skip layers, even if enough bandwidth is available. For the steady state, this is not a problem, but for practical implementation users might be impatient and would prefer to enjoy the first minute of a movie in HD if enough bandwidth is provided. In many deployed adaptive video players, during the initial loading of the video, bandwidth-based adaptation is used to quickly reach the highest quality layer that can be replayed with the given bandwidth, as is also proposed in [62]. This also includes that our model does not consider quality switches that

skip a layer, i.e., it can only choose either the next lower layer of the next higher layer.

3.2.2 Model Description

We model the buffer of the video player as a Markov queueing model as depicted in Figure 3.5. A video consists of segments. Each segment of the video is played in one of m layers. Segments arrive at the buffer with rate λ_i. Segments are removed (i.e. replayed) from the buffer with rate μ. When a segment of layer i is currently being downloaded, we say that the buffer is on layer i. Thus, the state of the system is defined by the number of Segments in the queue and by the layer on which the next segment is downloaded. E.g. the state $(0, 1)$ means that there is currently one segment in the buffer (which is currently being replayed) and a segment from the lowest quality layer is currently being downloaded. If the number of Segments contained in the buffer surpasses u_i, while the buffer is on layer i, then the buffer state switches from i to $i + 1$. If the number of Segments contained in the buffer underpasses l_i, while the buffer is on layer i, the layer i segment that is currently being downloaded will be discarded and instead the same segment will be downloaded on layer $i - 1$. This means that the buffer state switches from i to $i - 1$. There is a finite number of layers. The maximum number of segments is n on layer m. From this definition, the equations of the Markov model can be derived for the steady state, see Figure 3.1.

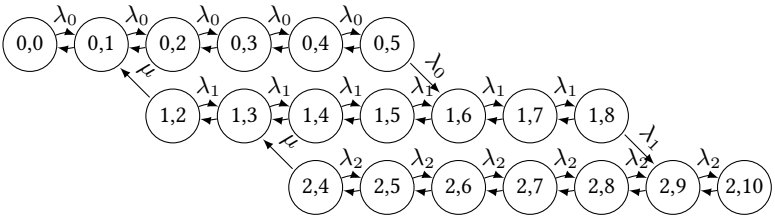


Figure 3.5: Markov chain for a buffer-based adaptation strategy of video streaming with three quality levels and thresholds $l_1 = 0; l_2 = 2; l_3 = 4; u_1 = 5; u_2 = 8; u_3 = 10$. All transitions from right to left happen with rate μ.

Table 3.2: Notations and Variables

parameter	description
λ_i	mean rate of segment arrivals while downloading a layer i segment
μ	mean rate of segment departures
i	quality layer of the segment that will arrive next, i.e. that is currently being downloaded
m_{max}	total number of quality levels
m	number of segments contained in the buffer/queue
n	maximum number of segments that reside in the buffer simultaneously
l_i	lower adaptation threshold for layer i: if *less* than l_i segments are in the buffer switch from layer i to layer $i-1$. l_1 is defined as 0.
u_i	upper adaptation threshold for layer i: if *more* than u_i segments are in the buffer switch from layer i to layer $i+1$. It is $u_m = n$.
$x(i,j)$	state probability that j frames are in the buffer and that the last quality level that was downloaded is quality level i

$$
\begin{aligned}
\lambda_1 x(0,0) &= \mu x(0,1) \\
\lambda_i x(i,j-1) + \mu x(i,j+1) &= \lambda_i x(i,j) + \mu x(i,j) \\
&\text{for} \quad i \in [0,m-1], j \in [1,n-1] \setminus \{l_i, l_{i+1}-1, u_i, u_{i-1}+1\} \\
\lambda_i x(i,l_{i+1}-1) + \mu x(i,l_{i+1}+1) &= \lambda_i x(i,l_{i+1}) + \mu x(i,l_{i+1}) + \mu x(i+1,l_{i+1}+1) && \text{for} \quad i \in [0,m-2] \\
\lambda_i x(i,u_i-1) &= \lambda_i x(i,u_i) + \mu x(i,u_i) && \text{for} \quad i \in [0,m-2] \\
\mu x(i,l_i+2) &= \lambda_i x(i,l_i+1) + \mu x(i,l_i+1) && \text{for} \quad i \in [1,m-1] \\
\lambda_i x(i,u_{i-1}+1) + \mu x(i,u_{i-1}+1) &= \lambda_{i-1} x(i-1,u_{i-1}) + \lambda_i x(i,u_{i-1}) + \mu x(i,u_{i-1}+2) && \text{for} \quad i \in [1,m-1] \\
\lambda_m x(m,n-1) &= \mu x(m,n) \\
\Sigma_{i=0}^{m} \Sigma_{j=0}^{n} x(i,j) &= 1
\end{aligned}
$$

Stationary Equations 3.1: Equations for the stationary distribution of the $M/M/1$ Markov model for buffer-based adaptive streaming.

3.2.3 Derivation of Key Impact Parameters

In the following, we determine the key impact parameters of the Markov chain. The state $(0,0)$ is the only state in which no segments are currently in the buffer. Stalling cannot occur on any other layer since we model the current layer as the quality layer that is currently being downloaded. Thus, the ratio of stalling events is given by $R = x(0,0)$, i.e. the probability of being in state $(0,0)$. Frequency of stalling events is given by $N = x(0,1) \cdot \mu$, i.e. the frequency of reaching state $(0,0)$. The duration of stalling events is $L = \frac{1}{\lambda_1}$. The duration of stalling events is equal to the initial delay since both only occur in state $(0,0)$. The rate of switching from a layer i to a higher layer $i+1$ is $S_{i,up} = x(i,u_i) \cdot \lambda_i$ for $i \in [1, m-1]$. Conversely, the rate of switching from a layer i to a lower layer $i-1$ is $S_{i,down} = x(i,l_i) \cdot \mu_i$ for $i \in [2,m]$. Thus, the total rate of switching from one layer to another is given as

$$S = \sum_{i=0}^{m-2} S_{i,up} + \sum_{i=1}^{m-1} S_{i,down} \tag{3.2}$$

$$= \sum_{i=0}^{m-2} x(i,u_i) \cdot \lambda_i + \sum_{i=1}^{m-1} x(i,l_i) \cdot \mu_i \tag{3.3}$$

$$= \lambda_1 + \mu_m + \sum_{i=1}^{m-2} x(i,u_i) \cdot \lambda_i + x(i,l_i) \cdot \mu_i. \tag{3.4}$$

A layer i segment is downloaded with rate

$$\nu_i = \sum_{j=l_i}^{u_i} x(i,j) \cdot \lambda_i. \tag{3.5}$$

Therefore, the mean layer on which a video is downloaded/replayed is given by

$$X = \frac{\sum_{i=0}^{m-1} i \cdot \nu_i}{\sum_{i=0}^{m-1} \nu_i}. \tag{3.6}$$

The mean bit rate on which the video is played BR is based on the mean bit rate of each layer BR_i as follows

$$BR = \frac{\sum_{i=0}^{m-1} BR_i \cdot \nu_i}{\sum_{i=0}^{m-1} \nu_i}. \tag{3.7}$$

3.2.4 Impact of Layer Distribution on Stalling

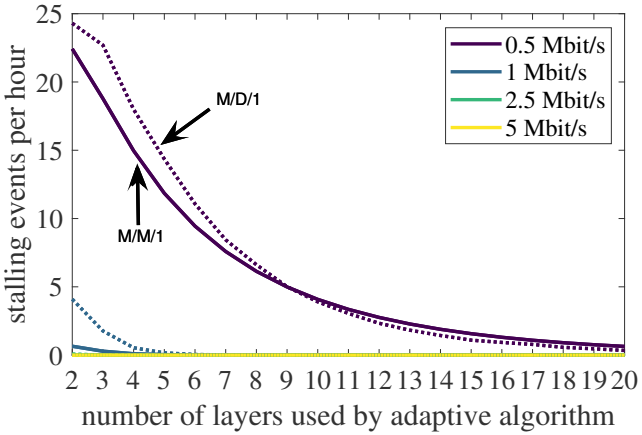


Figure 3.6: Impact of number of layers on the number of stalling events per hour for four typical throughput scenarios. Maximum buffer size was set to 90 segments (corresponding to 180 *seconds)*

In this section we investigate the impact of the number of layers on the number of stalling events. For this, we consider up to 20 quality layers. We set the lower switching thresholds to $l_i = \{0, 4, 8, 12, ...\}$ segments and the upper threshold to $u_i = \{10, 16, 22, 28, ...\}$ segments. The highest thresholds were set to 90 segments so that we always have the same maximum buffer. Segments have an

average duration of 2 seconds. As bit rates of the quality layers, we use values between $0.1\,\mathrm{Mbit/s}$ and $5.0\,\mathrm{Mbit/s}$ on a logarithmic scale.

In Figure 3.6, we observe that stalling events only occur for network bitrates below $1\,\mathrm{Mbit/s}$. Furthermore, we notice that a high number of layers leads to a much lower number of stalling events. In a low throughput scenario, 10 instead of 2 layers reduce the stalling frequency by over 70%. However, the number of layers does not have an impact on the mean played bit rate. The downside of a high number of layers is the increased data that needs to be stored at the video hosting provider and the increased encoding effort.

3.3 Optimization Potential from Elimination of Redundant Traffic

Next, we take a closer look at the behavior of the adaptation logic of common video players. In previous work [125] it was shown that an adaptation logic that was used by YouTube focuses strictly on the user, at the expense of network efficiency. In particular, we observed that the YouTube player sometimes discarded its currently buffered content to re-download it in a higher quality level. In this way, the player can increase the average quality level shown to the user. However, the overall efficiency decreases as the previously downloaded segments are discarded. At the example of an experimental data set, we analyze how much the used adaptation algorithm can be optimized. Even if we completely avoid stalling events, a higher mean video quality is achievable in most cases. Further, it is possible to reduce the number of resolution switches and start the video after a shorter initial delay.

The evaluation in this work is based on an experimental data set with over 10.000 video views of about 30 different videos. The videos were played in a testbed where the connection was throttled to $\{0.4, 0.5, \ldots 3.0\,\mathrm{Mbps}\}$. A proxy was used to decrypt the HTTPS connection. The data set and testbed is described in detail in [125, 157] and the experimental data set is freely available online at

[158]. Over 70 QoE-relevant metrics such as average quality level, cumulative stalling times and number of quality switches were collected.

3.3.1 Methodology

In this section we discuss the two approaches we use to evaluate the observed adaptation from the experimental data set. This first approach is based on regression and uses previously observed video sessions to create an estimation on how much non-redundant traffic relates to a specific average playback quality. This has the advantage of being fast, scalable and not computationally expensive. Furthermore, as it is based on actual observed data, it captures the dynamics of the deployed system. We use this estimation then to calculate the maximum achievable average quality level based on the total amount of downloaded Bytes in a playback session. The second approach is uses an integer linear program (ILP) formulation. For this optimization problem we take the actual video segment sizes, the observed bandwidth and cumulative stalling times from the experimental data set as an input. This gives us the optimal adaptation considering the stalling times. In a second step, we remove the cumulative stalling times and force the optimization problem to instantly play the video.

Heuristic Approach

To describe the heuristic, we first have to define redundant traffic. The redundant traffic ratio is defined as in the subsequent equation, where B_T is the total amount of data downloaded during the playback session and B is the sum of the segments' sizes shown to the user.

$$\rho = \frac{B_T - B}{B} \tag{3.8}$$

The heuristic approach uses isotonic regression [159] to deduce a video-dependent relationship between the data shown to the user and the resulting average quality level based on previously recorded playback sessions. This gives

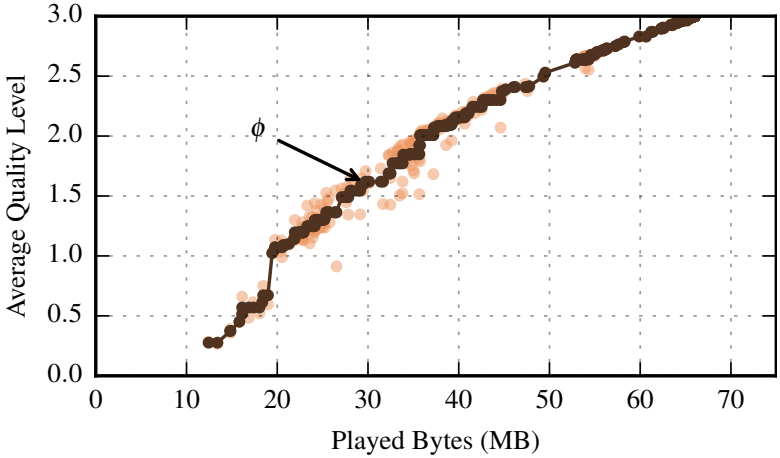


Figure 3.7: Isotonic regression result showing the relationship between Bytes shown to the user B and resulting average playback quality for video vbLLqaa9ksw. 336 video views are used in this regression.

an estimate of how much non-redundant data is necessary to reach a certain quality level. Furthermore, it allows us to estimate the difference in terms of average quality between two different amounts of data. The advantage of the approach is, as previously described, that it captures the dynamics of the overall system as it is based on actual observations.

Let $\phi(B)$ be the functional relationship between the quality level ϕ and the Bytes B. Figure 3.7 illustrates the function ϕ for one of the videos in the data set. The x-axis gives the amount of Bytes B played back by the player. The y-axis gives the resulting average playback quality. Each (brown) dot represents one playback session. The connected (black) dots are the isotonic regression result. Multiple observations can be made from the figure. First, a specific amount of played bytes can result in different average quality levels at the end. This is due to the combinatorial problem which arises due to the different quality levels and

bit rate variations inside a quality level. Second, there is a jump at 20 MB from 0.7 to 1.1 average quality level of unknown origin. Third, there are outliers, e.g. at 27 MB, where significant more data does not increase the average quality level.

Based on ϕ we determine the loss in average quality level, or *possible gain*, due to the redundant traffic as:

$$\phi(B_T) - \phi(B) \tag{3.9}$$

This is the difference between the average quality level we could have reached with the total Bytes downloaded in the session ($\phi(B_T)$) and the average quality level based on the Bytes shown to the user $\phi(B)$.

Optimal Adaptation

In order to determine how much potential there is for optimization, we use a modified version of an ILP provided in [114]. The solution to the ILP will return an optimal adaptation with respect to available bandwidth, video segment sizes, and cumulative stalling times.

A given video is available in r resolutions and consists of n segments, i.e. each segment can be played in exactly one resolution. Furthermore, each segment i that is played in resolution j has a size S_{ij}. We assume that all segments have the same duration τ and are downloaded in order. The total data that has been downloaded at the point in time t is $V(t)$. Before a segment can be played, it has to be downloaded. This means there is a deadline D_i until which the segment must be downloaded to avoid stalling. Since there is an initial delay T_0 before the first segment can be played, according to [97] the deadline is

$$D_i = T_0 + i \cdot \tau. \tag{3.10}$$

The goal is to optimize the downloading process so that the video may be played with the highest average resolution. This leads us to an ILP which is a special case of *OP1* from [114] and *Optimization Problem 2* from [97].

$$
\begin{aligned}
\text{maximize} \quad & \sum_{i=1}^{n} \sum_{j=1}^{r_{\max}} j x_{ij} \\
\text{subject to} \quad & x_{ij} \in \{0,1\} \\
& \sum_{j=1}^{r_{\max}} x_{ij} = 1, && \forall i = 1, \ldots, n \\
& \sum_{i=1}^{k} \sum_{j=1}^{r_{\max}} S_{ij} x_{ij} \leq V(D_k), && \forall k = 1, \ldots, n.
\end{aligned}
$$

This is a Multiple-Choice Nested Knapsack Problem which is NP-hard. However, there exist polynomial time algorithms that return an approximation for the optimal solution that is sufficiently good for most practical purposes. The ILP was implemented in Gurobi[2] in MATLAB.

Data Sets

In total, there are four data sets used in this evaluation as listed in Table 3.3. The three data sets starting from the second are calculated based on the first (experimental) data set.

First, we have the initial observations which shall serve as the *baseline* in the following analysis. These measurements were originally recorded in [125] where the measurement methodology and measurement set-up is described in more detail: 35 videos $\times 27$ bandwidth values $\times 15$ replications. Four quality level representations were observed: $144p, 240p, 360p, 480p$. In the following,

[2] http://www.gurobi.com/

Table 3.3: Overview of the data sets used in Section 3.3.2.

Data Set	Identifier	Description
Measurements from [125]	measurement	The experimental data set recorded in a testbed.
Heuristic estimation	heuristic	The heuristic estimation which gives us the possible gain without redundant traffic.
Optimization with stalling	opt (prebuffering)	The ILP solition with stalling times.
Opimtization without stalling	opt (instant play)	The ILP solution without stalling times.

we refer to these video quality levels as $0, 1, 2, 3$. Please note that stalling events did occur in 56% of these runs.

Based on this data set, we used the *heuristic approach* described in 3.3.1 to estimate the average resolution that is reachable if there was no redundant traffic, i.e. when no video segment is downloaded multiple times. Please note that it was assumed that the same amount of stalling would occur.

As a new contribution, we use the optimization problem, described in Section 3.3.1 to exactly calculate the highest mean resolution that was optimally obtainable. As a second step, the number of switches is minimized as first proposed in [114]. For both steps, we limit the execution time of the Gurobi Optimizer to $1\,\mathrm{s}$ in order to process the complete data in a timely manner. Increasing the execution will most likely lead to slightly better values than presented in the following. For this *two-step approach*, we consider the same video files, the same duration of the viewing session and the same average network throughput as was used in the baseline scenario to make it comparable. However, instead of having stalling events interrupt the replaying process, we add an initial delay to the replaying process. The duration of this delay is equal to the sum of the observed stalling events. This leads to the same duration of the viewing session and the

same replay time and the same amount of data that was totally downloaded. In the following, we refer to this scenario as *opt (prebuffering)*.

Lastly, we present a data set that is obtained in the same fashion as *opt (prebuffering)* with one major difference: the video starts to play immediately after the first segment has been downloaded. To achieve this, we consider the exact same network throughput as in the baseline scenario, while having a shorter session duration since the stalling times are omitted. This means that the amount of data that is downloaded in this case is lower than in the baseline scenario. In the following, we refer to this scenario as *opt (instant play)*.

3.3.2 Results

In this section, we present our results on how much YouTube's current adaptation algorithm could be improved. As key metrics, we analyze the average quality, the frequency of stalling events, the frequency of quality switches and the initial delay of the video playback. As a first step, we discuss how the video quality and stalling events are related to each other in the experimental data set.

Next, we discuss the potential gain in average quality as estimated by the heuristic and the two optimization problem formulations. Figure 3.8 displays the distribution of the difference between the observed mean video quality and the optimally achievable mean video quality. We observe that about 30 percent of runs are already at maximum quality and can therefore not be improved. The data set *opt (prebuffering)* leads to the highest mean quality. However, the results for the three data sets are very close to each other, e.g. the median of all three data sets is within 0.15 of a quality of each other.

If we take Figure 3.9(a) into consideration, it becomes clear that this minor difference in quality comes at a price: *opt (instant play)* demonstrates that it would have been possible to avoid stalling and a high initial delay in 93 percent of cases while increasing the quality in 30 percent of cases. While *opt (prebuffering)* shows that the mean quality could have been increased by adding an initial delay, the improvement is not particularly high. Lastly, while the heuristic leads to a worse

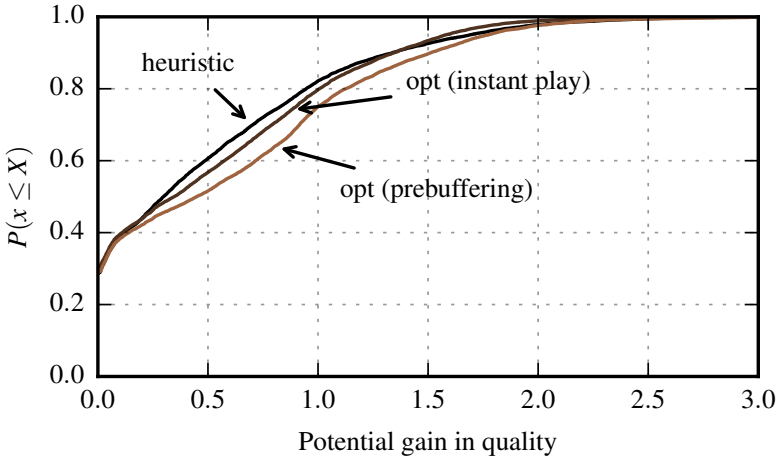
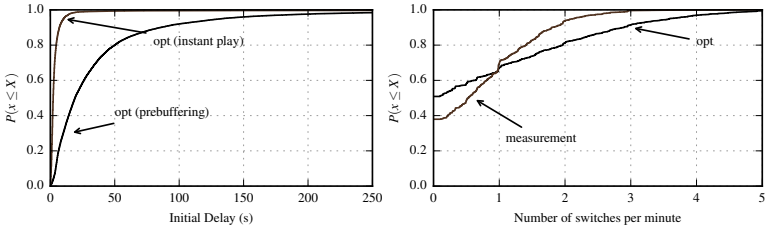


Figure 3.8: Distribution of the difference between the observed mean video quality and the optimally achievable mean video quality according to the optimization problem from Section 3.3.1 and the heuristic from [125].

result than *opt (prebuffering)*, it has the advantage of being a less complex problem. This might outweigh the slightly better performance for practical purposes.



(a) Distribution of the initial delay for the two optimization data sets.

(b) Switches per minute for the heuristic and the optimization. Very similar results for both optimization data sets.

Figure 3.9: Investigation of optimal adaptation.

Finally, Figure 3.9(b) shows the number of quality switches per minute. Here, both data sets that were created with the optimization approach lead to very similar results, which is why we only present the results for *opt (instant play)*. Whereas the number of switches is not of significant importance to the QoE in video streaming according to [56], continuous video quality switches lead to a low QoE [160]. The heuristic approach and the optimization both lead to less than 2 switches per minute in more than 80 percent of cases which are acceptable values. However, the two-step approach for the optimization leads to some very high switching frequencies that might be problematic. This is because the two-step approach puts very high value on the optimization of the quality level and very little emphasis on the number of switches. Luckily, this problem can easily be averted by using a slightly different approach: In [9], a method is proposed that combines both steps into one, allowing the number of switches to be emphasized higher at a negligibly low cost of quality.

3.4 The Trade-Off Between Quality and Switches

In the next section, we investigate the trade-off between average video quality and quality switches with regard to a weighting parameter. For this purpose, we use a quadratic program that optimizes adaptation in video streaming towards higher quality and fewer quality switches. This is done with respect for the users individual preference for these two QoE impact factors. Our research question can be formulated as follows. *What is the trade-of between the average quality of a video and the number of quality switches in adaptive video streaming?*

We demonstrate the optimization of the adaptation with a real mobile goodput trace and 41 different YouTube videos. Various user preferences for the adaptation are respected by conducting a parameter study for the adaptation aggressiveness parameter α that defines how frequently the player may switch to another quality. We then investigate the resulting optimal adaptation paths and evaluate key QoE indicators, such as the switching frequency, the average video quality and the buffer level.

3.4.1 Problem Formulation

In the following, we present an exact formulation of the problem that we want to optimize in this Section. Consider a video that consists of n segments. Each segment i is downloaded in exactly one of r quality layers. In order to play a video without stalling, each segment i must be downloaded before its deadline D_i. The data that is available from the initial video request at a point in time t is defined as $V(t)$. The initial startup delay is fixed to 5 s in the evaluation, i.e. $V(0)$ equals the sum of the goodput of the first 5 s. The size of segment i on layer j is S_{ij} and is given in Byte. If two consecutive segments are downloaded on different layers, the viewer experiences a quality switch. In order to maximize the viewers quality of experience, we want to increase the mean quality and reduce the number of quality switches. The importance of these two parameters is set as $\alpha \in [0, 1]$. Higher values for α indicate that it is more important to avoid quality switches than to increase the average quality. Different users have

different preferences in this regard and thus different values for α. If a segment i is downloaded on layer j, then we define $x_{ij} := 1$, otherwise $x_{ij} := 0$.

The goal is to decide, on which quality layer we must download each segment in order to maximize the weighted sum of mean quality and the number of quality switches while avoiding stalling. This optimization problem can be formulated as a quadratic program as follows.

$$\text{maximize} \quad \sum_{j=1}^{r} \left(\frac{\alpha}{nr} \sum_{i=1}^{n} j x_{ij} - \frac{1-\alpha}{2(n-1)} \sum_{i=1}^{n-1} (x_{ij} - x_{i+1,j})^2 \right) \tag{3.11}$$

$$\text{subject to} \quad \sum_{j=1}^{r} x_{ij} = 1 \;\; \forall i \in \{1, \ldots, n\} \tag{3.12}$$

$$x_{ij} \in \{0, 1\} \;\; \forall i \in \{1, \ldots, n\}, \; j \in \{1, \ldots, r\} \tag{3.13}$$

$$\sum_{i=1}^{k} \sum_{j=1}^{r} S_{ij} x_{ij} \leq V(D_k) \;\; \forall k \in \{1, \ldots, n\} \tag{3.14}$$

The objective function (Equation 3.11) maximizes the weighted sum of the mean quality and the number of quality switches. In order to receive values between 0 and 1, we normalize the mean quality by the maximum quality r, we normalize the switches by the highest possible number of switches $n - 1$ and we add the factor $1/2$ to the quadratic term since it increases by 2 with every switch. Constraint 3.12 and 3.13 ensure that each segment is download in exactly one quality. Constraint 3.14 ensures that each segment k is downloaded before its deadline D_k while not more data than $V(D_k)$ is downloaded.

Further constraints can be added to such a problem based on the scenario. For example, in a mobile scenario, where a sudden loss of connectivity could occur, we could define a set of coverage zones Q_{zone}, where the bandwidth is set to 0. A blind spot (l, m) starts at segment l and ends at m. In order to avoid video stalling in the blind spot, the video needs to be downloaded up until segment

m. This can be expressed by adding the following constraint to the quadratic program:

$$\sum_{i=1}^{m} \Delta S_i x_i \leq \Delta V(D_l), \forall (l, m) \in Q_{\text{zone}}.$$

3.4.2 Evaluation

Under the assumption that the player does not do breaks in between segment downloads and that the player buffer is unlimited, the following holds true. An aggressive switching behavior results in a high average quality, high number of switches and a low average buffer level. A conservative switching behavior decreases the average quality, decreases the number of switches and increases the average buffer level in the player. From this it follows, that the main objectives of a QoE-aware streaming player, i.e. to increase the average quality, to decrease the number of switches and to avoid stalling by keeping the buffer level high, are contradictory. The main question of the evaluation is: *Can we keep the average quality high while at the same time reduce the number of switches and increase the average buffer level?*

For the evaluation, we use a challenging mobile scenario that was recorded in [96]. The original trace is scaled to a mean of 0.33 Mbps, 0.67 Mbps and 1.34 Mbps, while keeping the coefficient of variation the same (0.38). We denote the resulting three patterns as *low*, *medium* and *high*. Furthermore, we define seven shifted versions of the patterns where we move the starting point forward and append the skipped goodput samples to the end of the patterns. The starting timestamps are {0 s, 120 s, ..., 720 s}. Thus, we use in total $7 \cdot 3 = 21$ goodput patterns in the evaluation for each video sequence. We use 41 videos that represent the content-mix of YouTube videos. Based on the downloaded video files, we split the videos in segments with a duration of 5 seconds and use the segment sizes as input for the optimization. For details about the video selection process we refer the reader to a previous study [125] where the same videos were used.

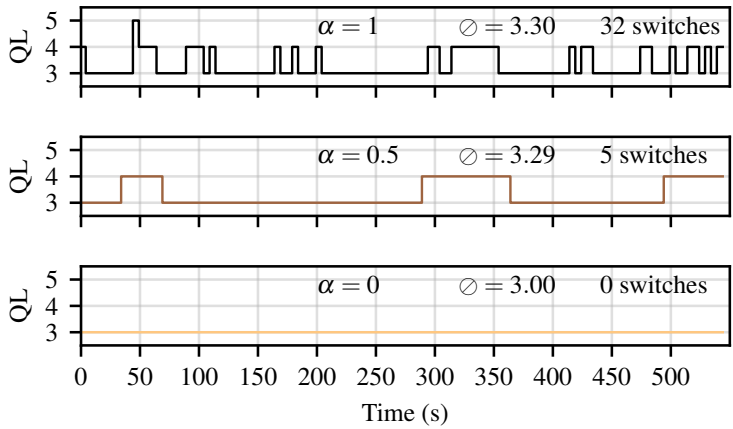
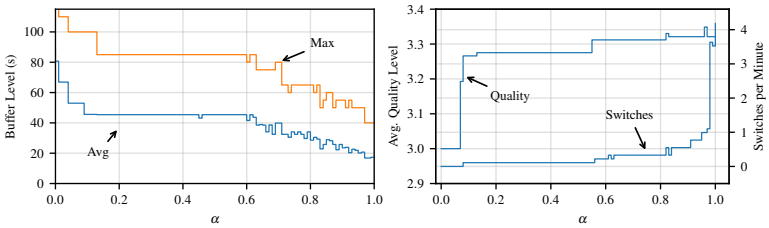


Figure 3.10: Adaptation path for three different values of α for the example video CRZbG73SX3s, medium goodput pattern and a starting timestamp of 0 s. $\oslash$ denotes the average quality. An increase in α increases the switches and the quality. From $\alpha = 0.5$ to $\alpha = 1.0$ the increase in quality is only 0.01 quality levels, while the switching frequency increases dramatically.

In the following we first discuss the influence of the α parameter by example. Figure 3.10 illustrates the adaptation path for three different values of α for the video CRZbG73SX3s under the medium traffic pattern and a starting timestamp of 0 s. $\oslash$ denotes the average quality of the playback. It can be observed that for $\alpha = 0$ the adaptation path is very conservative as there are zero quality switches and quality level 3 is selected from start until the end of the playback. For $\alpha = 0.5$, the number of switches increases to five and the average quality to 3.29 as the adaptation path is able to show for three periods during the playback a higher quality level. For $\alpha = 1$, the adaptation path is very aggressive with 32 quality switches, i.e. about 3.5 per minute, and one period where even quality level 5 is selected. However, the average quality increase is marginal with 0.01 compared



(a) Average and maximum buffer level. Aggressive choices of α decreases the observed average buffer level compared with very conservative choices. Results are presented as median over all seven starting timestamps.

(b) Average playback quality versus switching frequency. Starting from $\alpha = 0.1$ a more aggressive switching strategy does not result in increasing average playback quality.

Figure 3.11: Impact of α for on the playout of video CRZbG73SX3s using the medium goodput pattern.

with five switches for $\alpha = 0.5$. It follows that in this example the α parameter is able to adjust the aggressiveness of the adaptation path. Furthermore, we see that a high number of switches is not necessarily helpful in increasing the average quality.

Observations for example video CRZbG73SX3s

Next, we illustrate the relationship between the choice of α and the average and maximum buffer level by example. Figure 3.11(a) shows the average and maximum buffer level in seconds for video CRZbG73SX3s for the medium goodput pattern. The average buffer level is the time-dependent average over the buffer level values observed during the playback. The maximum is the highest buffer level observed during playback.

Two major observations can be made from the figure. First, the buffer level decreases for more aggressive values of α. For a conservative choice of alpha, e.g. $\alpha = 0.05$, the buffer level is on average about 53 s and maximum 100 s. For an aggressive choice, e.g. $\alpha = 1.0$, the buffer level is only 17 s on average and a

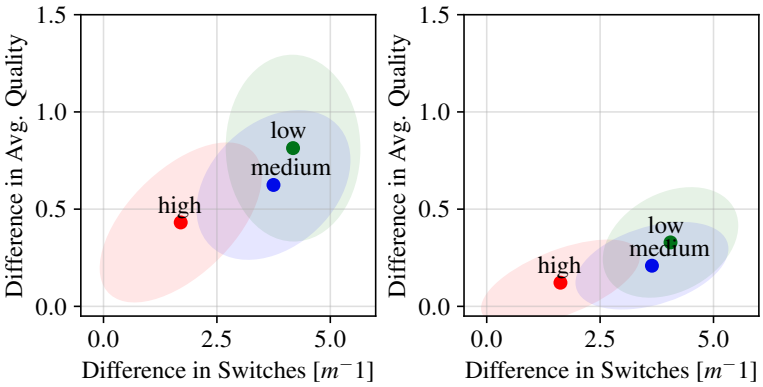
maximum of 40 s is observed. The second major observation is the fact that the buffer level on average is around 40 s, for the conservative switching behavior as well as more aggressive values up to $\alpha = 0.6$. Larger buffer levels reduce the risk of stalling events due to wrong adaptation decisions. From this it follows that an adaptation logic can prefer higher average quality and still keep a comfortable buffer level during playback.

Subsequently, we take a look at the tradeoff between the average quality and the switching frequency as the median of the different starting timestamps for the video CRZbG73SX3s for the medium goodput pattern. Figure 3.11(b) shows the average playback quality (left axis) and the switching frequency (right axis) for different values of α.

For $\alpha \leq 0.07$, the number of switches is zero and the average playback quality is 3.0, as also observed in Figure 3.10. For values of α between 0.07 and 0.55, the switching frequency increases to 0.1 switches per minute and the quality increases rapidly to 3.28. The difference in average quality compared to Figure 3.10 is due to the fact that we consider here all starting timestamps of the goodput pattern, while Figure 3.10 shows only one particular. Starting from $\alpha = 0.6$ to $\alpha = 0.94$, the switching frequency increases up to 0.8 switches per minute, while the average quality stagnates at around 3.31. If α is further increased, the switching rate increases rapidly up to 4.2 switches per minute while the average quality only increases marginally to 3.32. This example is in line with the previous observations that a more aggressive switching frequency does not necessarily benefit the average playback quality. In contrary, the experience of the user is diminished by frequent quality switches while on average the playback quality cannot be increased by the frequent switches.

Observations for all videos

Next, we evaluate the following question by looking at the whole set of videos. *What is the maximum achievable gain in terms of average playback quality when using an aggressive switching strategy compared with a conservative one?* Figure 3.12 presents the difference in switches rate and difference in average playback



(a) $\alpha = 0.01$ (b) $\alpha = 0.1$

Figure 3.12: Difference in average quality and switching rate between $\alpha = 1.0$ and $\alpha = \{0.01, 0.1\}$ for the three goodput patterns low, medium and high. The shaded areas denote the two-dimensional standard deviation of the samples.

quality between $\alpha = 0.01$ and $\alpha = 1.0$ (Fig. 3.12(a)) and between $\alpha = 0.1$ and $\alpha = 1.0$ (Fig. 3.12(b)) for the three goodput patterns over all 41 videos. The shaded areas denote the 2 dimensional standard deviation of the samples. The dots represent the mean of the samples. Multiple conclusions can be drawn from this figure. First, the lowest gain in quality can be observed for the high goodput pattern. This is due to the fact that there are many videos in the set where the high goodput pattern offers enough traffic volume to download always the highest quality level. There are no switches needed for those videos. Second, the mean of the low goodput pattern reaches 4.6 m^{-1} switching rate and a difference of 0.8 quality level for $\alpha = 0.01$. This means that on average in a low goodput scenario you can improve the average quality level by 0.8 by aggressively switching the quality level on average every 13 s. Third, the difference in average quality drops considerable when comparing $\alpha = 0.1$ with $\alpha = 1.0$. For example,

the maximum achievable gain for the low goodput pattern drops from 0.8 to about 0.4. Consequently, from the user's perspective there is only a small gain in switching with a higher aggressiveness than $\alpha = 0.1$.

3.4.3 Discussion

At first, we discuss the methodology and we propose a modification of an existing optimization formulation which allows us to calculate an optimal adaptation path for a given video, a given goodput pattern and a given switching-versus-quality trade-off parameter. This trade-off parameter defines the aggressiveness of the quality switching behavior and is denoted by α. Choosing $\alpha = 0$ results in an adaptation path with zero quality switches and $\alpha = 1$ results in an adaptation path which tries to optimize the average quality at all cost, i.e. with as many quality switches as necessary. Based on an example video we observe that the average buffer level drops fast for values of α between 0.0 and 0.1. For $\alpha > 0.1$, the average buffer level stays close to 42 s and starting from $\alpha = 0.6$ drops linearly to 20 s. Afterwards we take a look at the average playback quality and the switching frequency for the example video. The results show that the switching frequency increases with $\alpha \geq 0.6$ rapidly, while the average quality increases fast and reaches its maximum early at about $\alpha = 0.1$. An evaluation of 41 randomly selected videos from YouTube shows that on average an increase of up to one quality level is possible by increasing the switching frequency by 5 switches per minute. However, the evaluation also shows that this increase is half due to the sharp increase in average quality for $\alpha = 0.01$ to $\alpha = 0.1$.

In general, the results show that aggressive switching behavior is not necessary rewarded with a higher average playback quality. From the evaluation also follows that a good starting point for future evaluation of the α parameter is $\alpha = 0.1$. Higher, more aggressive, values do not increase the average playback quality much further.

Figure 3.13 illustrates qualitatively the application of our contribution to the future work in the area of HAS adaptation research. The optimization formula-

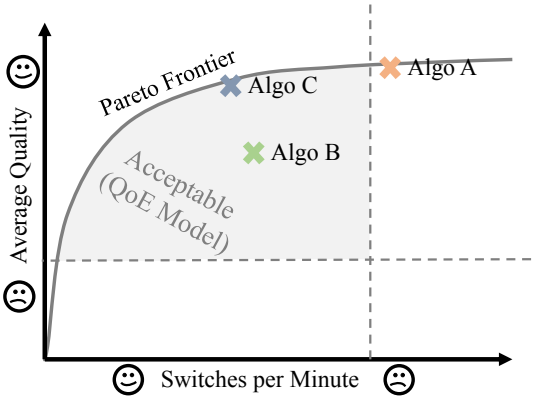


Figure 3.13: Application of our contribution for future work. The optimization formulation enables us to determine the Pareto frontier for any given HAS adaptation algorithm, a given goodput pattern and a given video in terms of the trade-off between average quality and switching frequency. In combination with a future sophisticated QoE model, this allows for a novel evaluation and classification of adaptation algorithms for HAS video streaming.

tion allows us to determine the Pareto frontier for any given HAS adaptation considering the trade-off between maximizing the average quality and minimizing the number of quality switches during playback. This means that no existing or future HAS adaptation can reach a higher average quality for a given number of switches than the Pareto frontier. Furthermore, we know from the evaluation that the Pareto frontier quickly saturates in terms of average quality. It is ongoing user-experience research to determine a model for the lower bound for the average playback quality and an upper bound for the switching frequency. In combination with such a future sophisticated QoE model, the Pareto frontier allows for a novel evaluation and classification of adaptation algorithms for HAS video streaming.

3.5 Optimal Fairness in Adaptive Streaming

As video streaming is such a widely used service, it often happens that multiple video players play at the same time while sharing the same Internet connection. When the bandwidth is limited, thus this connection forms a bottleneck, the players compete for bandwidth and have to adapt the video quality accordingly.

In addition to maximizing quality across multiple users, an inherent question is that of how to address the issue of fairness. There is a clear need to distinguish between QoS fairness (e.g., resulting in equal bitrate allocations) vs. QoE fairness, leading to equal *utility* among users [136]. While client-side bitrate adaptation algorithms may in a best-case scenario achieve flow-based fairness, this will rarely translate to session-level fairness (or QoE level fairness for a given session) [161]. Recent papers have argued that a QoS fair system is not necessarily QoE fair, e.g., [162, 163], given the lack of consideration of service QoE models. Such models specify the relationships between user-level QoE and various application-layer performance indicators (e.g., file loading times, video rebuffering) or influence factors such as device capabilities, context of use, network and system requirements, user preferences, etc.

Even though a simple adaptation logic can perform very well for a single user, in a scenario with multiple users the video players could over- and underestimate the bandwidth due to mutual reactions between the adaptation logic. We therefore investigate the behavior of multiple video players which share a bottleneck link with variable and fixed bandwidth. Therefore, we developed a testbed which allows to run multiple video players in parallel sharing a bottleneck. We configure the bottleneck with real mobile throughput traces that have a high variance to investigate the performance of the current state of HAS in challenging situations. Literature proposes such concepts to ensure fairness with coordinated control [132, 164]. However, in current implementations of HAS (like YouTube and Netflix), clients base their adaptation decisions on local information like the buffer status or the network throughput history [165].

The first contribution of this section is a user-centric analysis of fairness for several simultaneous HAS clients. In particular, we answer the following questions. (1) To what extent is stalling and the video quality affected when video streaming clients share a network bottleneck? (2) Is the simultaneous playback of multiple adaptive video players fair or is there a significant difference between them? To answer these questions, we conduct a measurement study where multiple users who share a bottleneck are watching videos at the same time over a long period. We investigate to what degree the video resolution and the frequency and duration of stalling events differ. We focus on QUIC, since it is used in Chromium / Chrome which is the most spread web browser. Our results show that there are fairness concerns with QUIC that are not observed in measurement studies that focus on TCP. The methodology of this measurement is described in detail in the Appendix A.

As a second contribution, we extend the approach from Section 3.4 by formulating a QoE optimal download strategy in the case of multiple HAS users accessing Video on Demand (VoD) content via a shared bottleneck link, i.e., users watch different videos at different starting points. A well-known issue linked to such a scenario is that the on/off nature of flows often results in inaccurate client-side bandwidth estimation and leads to potentially unfair resource demands, quality oscillations, and poor bandwidth utilization [129, 132, 162, 166]. While different approaches in literature propose methods to mitigate these problems by employing various monitoring and control solutions at different points along the service delivery path [162, 167, 168], what is missing is a methodology for comparing and benchmarking these different approaches. We thus propose a quadratic problem formulation to compute the theoretical optimum in terms of adaptation strategies and corresponding segment downloads across multiple users under given bandwidth constraints. We specify the objective as being to maximize average quality, minimize the number of quality switches, and ensure equal utility (QoE) among users while avoiding stalling events. We do this by extending the approach presented in Section 3.4.1 by a fairness component.

By aiming to maximize both service quality and fairness, we quantify and compare the impact of different fairness objectives (bandwidth fairness, pattern fairness, and session fairness) on resulting quality and achieved QoE fairness. Based on conducted simulations and parameter studies, our results demonstrate the benefits of optimizing for session fairness as compared with other approaches. quality switches.

3.5.1 Measurement Study on Fairness in HAS

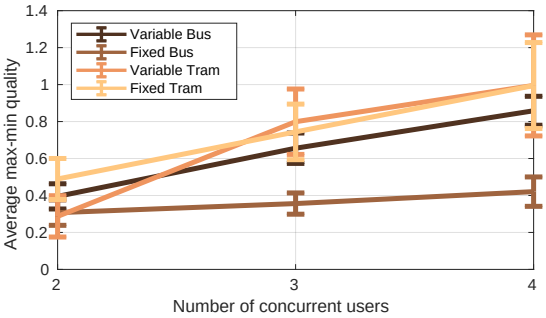


Figure 3.14: The average max-min quality of concurrent users.

In Figure 3.14 we compare the user with the highest average quality and the user with the lowest average quality during a single run. We call the difference between these two values the max-min quality. Notice that lower values indicate greater equality among the players. It can be observed that an increasing number of concurrent players leads to a greater max-min quality. The bus scenario which is characterized by a high average bandwidth and a low standard deviation has a significantly lower max-min quality than the other scenarios. For the tram scenario, a fixed bandwidth only leads to slightly higher inequality.

Finally, we look at the fairness in terms of the max-min number of stalling events in Figure 3.15. We first notice that a variable bandwidth leads to signifi-

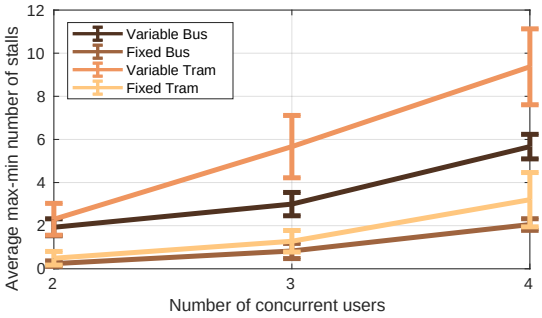


Figure 3.15: The average max-min number of stalling events of concurrent users.

cantly greater unfairness than fixed. In the tram scenarios there is also greater unfairness compared with the bus scenario due to the lower bandwidth which leads to an overall greater number of stalling events. The confidence intervals are given with a confidence level of 95%. From this, it follows that a combination of the YouTube algorithm and the QUIC protocoll is not fair over short periods of time for multiple users that share a bottle neck.

3.5.2 Optimization Problem

We assume a set of U users. Each user u downloads exactly one video Y_u. Each video is divided into n segments which must be downloaded in exactly one of r_{max} resolutions/layers[3]. The volume of segment i on layer j of video Y_u is defined as S_{uij}. Each segment i of video Y_u must be completely downloaded before its deadline D_{ui} to play the video without stalling. Each user u requests a video at a point in time T_u. We assume that users request videos sequentially, i.e., User 1 requests his video before User 2. The function $V(t_1, t_2)$ describes the

[3]If videos may have different numbers of layers or segments, we replace r_{max} with $r_{u,max}$ or n with n_u

Table 3.4: Notations and Variables

$u = 1, 2, 3 \ldots$	index for users (in order of requests)
U	number of users
Y_u	video that is downloaded/watched by user u
T_u	time of request of video Y_u by user u (second)
i	index for segments
n	number of segments
j	index for quality layer
r_{max}	number of quality layers
S_{uij}	size of segment i of video Y_u in quality j (Byte)
D_{ui}	deadline of segment i of video Y_u until which download must be completed (second)
$V(t_1, t_2)$	data that can be downloaded between the points in time t_1 and t_2 (second)
$x_{uij} \in \{0, 1\}$	solution whether segment i of video Y_u is downloaded in quality j or not
w_{uij}	weight for segment i of video Y_u in quality j, e.g. QoE value
F_{ui}	absolute fairness for the quality of segment i between user u and other users
F_u	absolute fairness for the mean quality of all segments between user u and other users
α	relative importance of the average video quality
β	relative importance of quality switches
γ	relative importance of quality fairness
δ	relative importance of upward switches compared with downward switches

data that can be downloaded between the points in time t_1 and t_2. For $V(0, t_1)$ we may use the shortened notation of $V(t_1)$.

$$\text{maximize} \quad \alpha \sum_{u=1}^{U} \sum_{i=1}^{n} \sum_{j=1}^{r_{max}} w_{uij} x_{uij} - \beta \sum_{u=1}^{U} \sum_{i=1}^{n} (\sum_{j=2}^{r_{max}} \sum_{k=1}^{j-1} (j-k) x_{uij} x_{u,i+1,k}$$

$$- \delta \sum_{j=1}^{r_{max}-1} \sum_{k=j}^{r_{max}} (k-j) x_{uij} x_{u,i+1,k}) - \gamma \frac{1}{nU} \sum_{u=1}^{U} F_u \tag{3.15}$$

$$\text{subject to} \quad x_{uij} \in \{0,1\} \quad \forall u = 1, ..., U, \quad \forall i = 1, ..., n, \quad \forall j = 1, ..., r_{max} \tag{3.16}$$

$$\sum_{j=1}^{r_{max}} x_{uij} = 1, \quad \forall u = 1, ..., U, \quad \forall i = 1, \ldots, n \tag{3.17}$$

$$\sum_{u=l}^{U} \sum_{i=1}^{k} \sum_{j=1}^{r_{max}} S_{uij} x_{uij} \leq V(T_l, D_{uk}), \quad \forall k = 1, \ldots, n, \quad \forall l = 1, \ldots, U \tag{3.18}$$

$$F_u \geq \sum_{i=1}^{n} \sum_{j=1}^{r_{max}} w_{uij} ((U-1) x_{uij} - \sum_{\tilde{u}=1, \tilde{u} \neq u}^{U} x_{\tilde{u}ij}), \quad \forall u = 1, \ldots, U \tag{3.19}$$

$$F_u \geq - \sum_{i=1}^{n} \sum_{j=1}^{r_{max}} w_{uij} ((U-1) x_{uij} - \sum_{\tilde{u}=1, \tilde{u} \neq u}^{U} x_{\tilde{u}ij}), \quad \forall u = 1, \ldots, U. \tag{3.20}$$

The optimization problem that we tackle can be formulated as follows: Optimize the average video quality of all users, while minimizing the number of downward quality switches and maximizing the number of upward switches. Quality switches are weighted with the difference in quality. Further, we minimize the difference between the average quality among users, while avoiding stalling events. The weight of each variable that is the subject of the optimization is defined by α for the average quality, β for the number of switches and γ for the fairness in quality among users. Upward switches may have a different impact on the user experience than downward switches. Therefore, we introduce a parameter δ that reflects the relative importance of upward switches compared with downward switches. We arbitrarily chose $\delta = 0.5$ for the remainder of this section. An overview of all parameters is given in Table 3.4.

Currently QoS fairness is employed via protocols that ensure that users who share the same link may use the same share of resources available. This means, we have fair network QoS. In the following we present two fairness schemes for application layer QoS.

Please note that in the quadratic program we do not use a fairness index such as Jain's or Hoßfeld's QoE fairness metric directly. The optimization problem formulates a theoretical implementation of a QoE management mechanism. Thereby, we rely on a very simple fairness measure for the sake of simplifying the quadratic program. For example, instead of relying on the standard deviation, we use the mean difference, avoiding squares in the formulation. More sophisticated fairness indexes such as the above can be applied to it nevertheless.

Session-Fairness

This approach corresponds to how fairness is evaluated over the whole session. Each user u has an average quality QoE_u in which he has viewed the video. The average quality QoE_u of each user should be as similar as possible to achieve high fairness.

The values α, β and γ are normalized so that the minimum and maximum value of their term is 0 and 1. Equation 3.17 means that each segment of any

video of any user must be downloaded in exactly one resolution. Equation 3.18 means that for each User l, the sizes of all segments, which are downloaded by users who requested a video after User l, may in their sum never exceed the data that can be downloaded since User l joined the system. Equation 3.19 and 3.20 are used to implement the absolute value for the difference to the mean. If $T_{\tilde{u}} - T_u < 0$ or $T_{\tilde{u}} - T_u > \max(T)$ then the constraint for this u is omitted. This has been left out of the equations for the sake of clarity. A downside of this approach for fairness is the difficulty of its implementation since we do not know whether users abandon videos early. A possibility to solve this problem is to implement a history-based fairness system, that considers all segments from the past, and tries to equalize the average quality of users over time. It is also possible to model this problem as a 2-step approach: In the first step, we ignore switches and determine the maximum quality and fairness W that is possible. In the second step, we try to reach at least $W - \epsilon$ and then minimize the number of switches. For the sake of brevity this approach is not presented in further detail.

Segment-Fairness

Another way to implement fairness for application layer QoS is to minimize the difference in quality between users for each video segment. This means that the nth segment watched by all users has similar quality, independently of when it is watched, e.g. in the case of maximum quality, the quality pattern in which videos are viewed is the same for each user. If we use segment-fairness (also referred to as pattern fairness), we have the following equations. We replace the last term of Equation 3.15 with

$$-\gamma \sum_{u=1}^{U} \sum_{i=1}^{n} \frac{1}{nU} F_{ui} \tag{3.21}$$

and Equations 3.19 and 3.20 with

$$F_{ui} \geq \sum_{j=1}^{r_{max}} w_{uij}((U-1)x_{uij} - \sum_{\tilde{u}=1}^{U} x_{\tilde{u},i+\frac{T_{\tilde{u}}-T_u}{\tau},j}), \tag{3.22}$$

$$\forall u = 1,\ldots,U, \quad \forall i = 1,\ldots,n$$

$$F_{ui} \geq -\sum_{j=1}^{r_{max}} w_{uij}((U-1)x_{uij} - \sum_{\tilde{u}=1}^{U} x_{\tilde{u},i+\frac{T_{\tilde{u}}-T_u}{\tau},j}), \tag{3.23}$$

$$\forall u = 1,\ldots,U, \quad \forall i = 1,\ldots,n$$

Bandwidth-Fairness

To determine the optimal result for the bandwidth-fairness, we first must determine, how much bandwidth is allocated to each user at which point in time. We assume that while n users are requesting segments or watching a video, $V_u = \frac{1}{n}$th of the bandwidth is reserved for each of them. Setting U to 1 and replacing V by V_u, we apply the optimization problem for each user to determine the optimal segment request sequence for each user. Notice, that by setting U to 1, Equations 3.19 and 3.20 are eliminated and the problem is simplified drastically.

3.5.3 Parameter Study for Optimization

In this section, we discuss the evaluation of the optimization program. We first discuss a sample run and then conduct parameter studies, to explain the impact of input values (i.e., scenarios) of the quadratic program.

Methodology

The scenario that we investigate in the following section is a low bandwidth scenario. Three users request a YouTube video (ID: CRZbG73SX3s, $\Delta t = 549s$) at different starting points $t_1 = 0s, t_2 = 245s, t_3 = 495$. Videos start playing after an initial delay of 5s. Each video is partitioned into video segments that have

an equal duration of $5s$ and are available in four quality levels $(1, 2, 3, 4)$ that differ in average bit rate: 144p (14 kBps), 240p (31 kBps), 360p (68 kBps) and 480p (127 kBps). The users share the same bottlenecked link with a constant effective bandwidth that we vary from 50 kBps to 150 kBps (indicated as $0.5, ...1.5$ in the figures).

We used a constant parameter $\alpha = 1$ which indicates that the video quality should always be of high importance. We varied the parameters $\beta, \gamma \in \{0, \dots 1\}$ to account for scenarios in which the degree of annoyance of quality switches and the degree of importance of fair video quality among users varies. We use $\delta = 0.5$ in every scenario. The quadratic programs were solved in Gurobi[4] with Matlab[5] as an interface. The program was executed on an i7 CPU with four cores with 2.70 GHz and 16 GB RAM. The run time of the program heavily depends on the scenario and the parameters used. The calculation for bandwidth fairness takes 0.2 s for $\beta = 0$ and 0.5 s for $\beta > 0$. The calculation for pattern fairness takes 5 s for $\beta = 0$ and 1700 s for $\beta > 0$. In the latter case we stop the program after 30 s and take non-optimal results to get results within an acceptable time frame. These results are very close to optimal results and do not differ visibly. For session fairness a run takes 0.8 s for $\beta = 0$ and 2 s for $\beta < 0$. The source code of the programs is available online[6]. We invite any scientists to use it for their own research. QoS fairness (also referred to as bandwidth fairness) was modeled by giving each user $\frac{1}{n}$ bandwidth while n users were in the network. Users always fully exhausted their available bandwidth. Then we optimized quality and switches according to α and β for the respective bandwidth pattern using Gurobi, ignoring QoE fairness. In Figure 3.16 an example of a single run is given in which we see the quality in which segments are watched over time for each user. The three metrics that we investigate are: average video quality, the fairness of the quality among users, and the number of quality switches.

[4] http://www.gurobi.com/

[5] https://www.mathworks.com/products/matlab.html

[6] https://github.com/ChristianMoldovan/Quadratic-Program-for-Optimal-Fairness-and-Quality-in-Video-Streaming-with-Multiple-Users

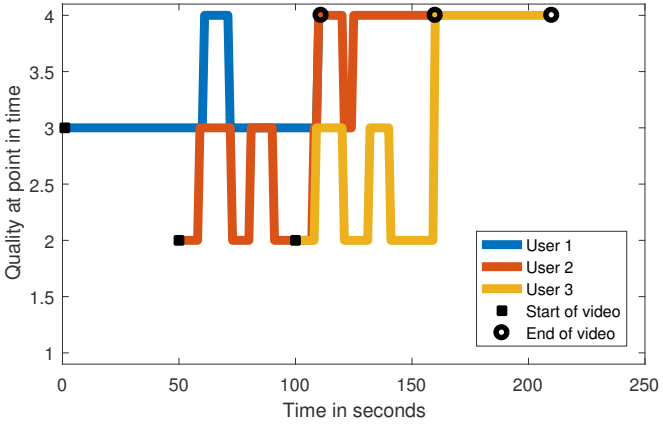


Figure 3.16: Quality of segments over time for three users watching a video with session-fairness employed and the following parameters: bandwidth $= 1.5$ *Mbit/s,* $\alpha = 1, \beta = 1, \gamma = 1$.

Trade-off Between Quality, Switches, and Fairness

As can be seen in Figure 3.17, the quality increases with the bandwidth, the number of switches is reduced with β, and fairness increases with γ. Furthermore, β and γ have no significant impact on the video quality. It is noticeable that the average quality is slightly lower when QoS fairness is enforced since there is less room for optimally scheduling the segment downloads. For example, when the third user joins the system in Figure 3.16, the user can only use $\frac{1}{3}$ of the available bandwidth since it is fairly shared with the other users. Therefore, User 3 can only download low quality segments. Optimally, when a new user starts a video, the user is given a larger share of the bandwidth initially, so they can start with a higher quality level. Other users may suffer with respect to obtained quality but switching from 240p to 360p is more cost effective as compared with going from 360p to 480p, if we consider the ratio of the bit rate to the quality difference. A

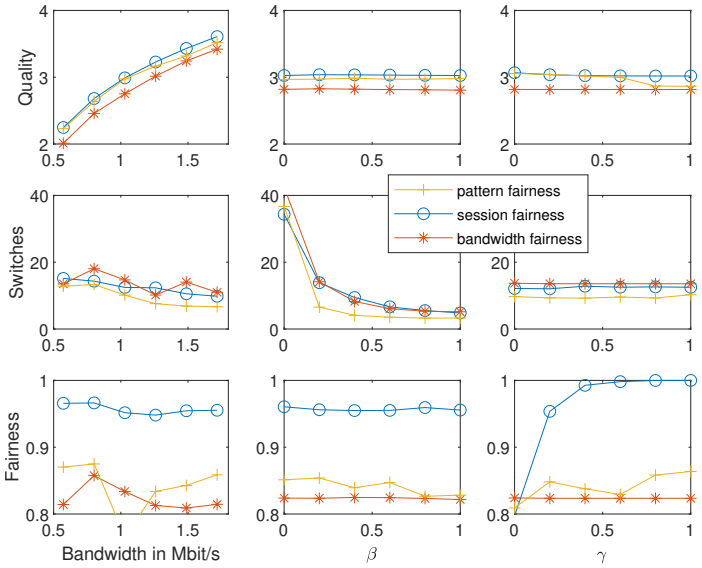


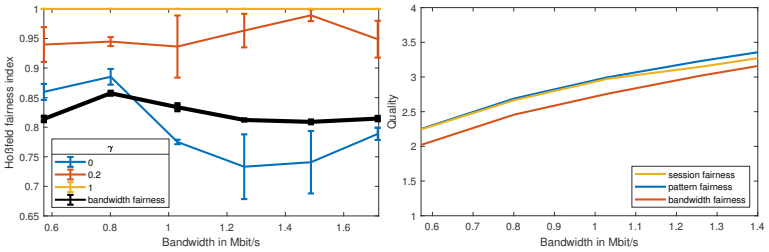
Figure 3.17: Impact of bandwidth, β and γ on the average video quality, the number of switches and the fairness of the video quality for $\alpha = 1$.

detailed analysis of the trade-off between quality and switches has already been conducted in [10].

In Figure 3.18(a) we see that a higher bandwidth leads to higher fairness. Furthermore, the fairness parameter γ leads to higher fairness, when approaching 0. In contrast, we can see the trade-off in Figure 3.18(b). While higher bandwidth also leads to higher quality, increasing γ leads to an increase in quality.

Bandwidth Fairness, Pattern Fairness, Session Fairness

Current systems mostly rely on network QoS fairness, due to its simplicity. Each user receives the same share of resources at any point in time. In the following,



(a) oS-Fairness: Videotrace number 1585. Impact of bandwidth and gamma on quality with $\beta = 0.4$.

(b) Session-fairness, segment-fairness and network QoS-fairness: impact of bandwidth on the video quality with $\beta = 0.4$. Mean over all γ values.

Figure 3.18: Impact of bandwidth on fairness and quality.

we emulate network QoS fairness by reserving $1/n$ of the total bandwidth for each user while n users are watching a video and are in the system.

With a network-QoS-fairly shared bandwidth, the average quality is slightly lower, compared with a perfect optimization. Even if perfect session fairness can be guaranteed, the mean quality of every single user is higher than with QoS-fairness. The fairness (in terms of Hoßfeld's fairness index) of the system is not impacted by γ, except for the simple case $\gamma = 0$ in which fairness is not considered in the optimization. This means that optimizing fairness for single segments has no impact on the overall fairness. Rather, it is impacted very much by the bandwidth. In contrast to Figure 3.18(a), fairness is always lower compared with optimizing session fairness. In Figure 3.18(b) we see that session-fairness leads to the highest quality with an average of 3.03, then segment-fairness with 2.92 and last network QoS fairness with 2.74. This means that the video quality can be increased by 0.29 quality layers in the above scenario on average, while resulting in fairer QoS on the application layer. While this might seem like a small value at first, it is unclear at what cost it comes in a practical scenario. If fairness schemes other than bandwidth fairness can be implemented at very low cost, even a small increase should not be disregarded.

3.6 Using Viewport Prediction for Optimal Adaptation in 360-Degree Videos

A straightforward approach to reduce the required resources for 360° streaming is to consider the viewport of the user, i.e., which part of the 360° sphere the user is currently focusing on. Tiles are streamed in the best possible quality only if they are in the focus of the user, and in a reduced quality if they are out of focus. If the viewport of the user would be known at any time during the streaming, an optimal adaptation strategy could then be obtained. In this section, we define and solve this optimization problem for omnidirectional videos in a given network scenario by an ILP. It can be used as a benchmark tool that is helpful to evaluate adaptation strategies for 360° videos.

3.6.1 Optimal Adaptation for 360-Degree Videos

We first define a baseline linear program for 360° video streaming, i.e., tile-based video streaming. This linear program computes the optimal adaptation strategy in a given network scenario if the viewport of the user would be known. As in practice the viewport cannot be known in advance, we extend the baseline program to take into account three approaches for viewport prediction. An overview of the three approaches is given in Figure 3.19.

Baseline Linear Program for Optimal 360-Degree Adaptation

In this problem, we download a set of n video segments in order. Each segment consists of m spatial tiles. To have a complete video, each tile of every segment must be downloaded exactly once, compare Eq. (3.25). Each tile can be downloaded in one of r resolutions or quality layers. The size in bytes of tile s of segment i in resolution j is defined as S_{ijs}. The quality of a segment is defined by its value function w_{ijs}.

For each segment i there exists a deadline D_i^2 until which it must be completely downloaded to allow seamless video streaming without stalling. The first segment

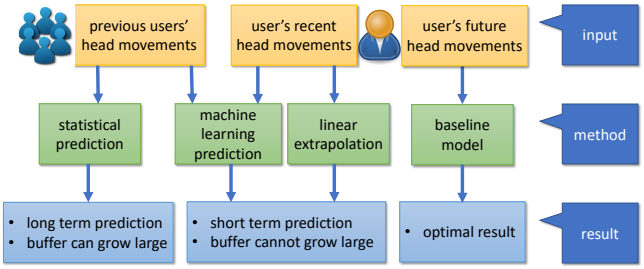


Figure 3.19: Overview of the four approaches with input and result. The baseline model only serves for comparison and is not applicable for practical purposes since it requires exact future knowledge.

must be completely downloaded within the allowed initial delay $D_1^2 = T_0$. The deadline of each consecutive segment i is defined as $D_i^2 = T_0 + (i-1) \cdot \tau, \quad \forall i > 1$, where τ is the duration of a segment.

The data volume that can be downloaded between time D_i^1 and D_i^2 is defined as $V(D_i^1, D_i^2)$. The sum of the volume of any consecutively downloaded tiles must not be greater than the available data volume until the last segments deadline, compare Eq. (3.27). In omnidirectional videos, the user only sees a subset of all tiles. Thus, we define the viewport Y_i of a segment i which includes all tiles of which at least one pixel is viewed during the segment. For the objective function only the viewed tiles are relevant. For the sake of simplification, we consider each tile equally important, even if it is viewed briefly or partially. The goal of the objective function (3.24) is to maximize the sum of the quality of all viewed tiles of all video segments while avoiding stalling. We formulate the corresponding binary linear program as follows:

$$\text{maximize} \quad \sum_{i=1}^{n} \sum_{j=1}^{r} \sum_{s \in Y_i} w_{ijs} x_{ijs} \tag{3.24}$$

$$\text{subject to} \quad \sum_{j=1}^{r} \sum_{s \in Y_i} x_{ijs} = 1 \tag{3.25}$$

$$\forall i \in \{1, \ldots, n\}$$

$$x_{ijs} \in \{0, 1\} \tag{3.26}$$

$$\forall i \in \{1 \ldots n\}, j \in \{1 \ldots r\}, s \in \{1 \ldots m\}$$

$$\sum_{i=1}^{k} \sum_{j=1}^{r} \sum_{s \in Y_i} S_{ijs} x_{ijs} \leq V(0, D_i^2) \tag{3.27}$$

$$\forall k \in \{1, \ldots, n\}$$

Optimal 360-Degree Adaptation under Viewport Prediction

In a practical use case, the viewport of the user is not known in advance. Thus, it is necessary to predict the viewport in the next seconds of the playback and rely on these predictions for adaptation decisions. Thus, we extend the baseline program to take into account three approaches for viewport prediction, which are also presented in Figure 3.19:

- a statistics-based approach that uses other users' viewports to determine the viewport probability for each tile during each video segment,
- a linear extrapolation of the head movement trajectory of the current user, and
- a deep neural network that trains with previous users' head movement sequences during the same video before being applied to the current user's trajectory.

Statistical Long-Term Prediction

The first method assumes that most users share a common interest in a subset of all tiles during a segment. The idea is to download tiles in a higher resolution if they are watched more frequently. Since this method is independent of the current viewport, we can download tiles early and fill the buffer. For this, we require the frequency P_{isu} that a tile s of segment i is viewed by a viewer u. We then determine the probability $P_{is} = E[P_{isu}]$ that a tile is viewed over all previous users. For any new user, we use this probability P as a weight for the objective function. Consequently, in the linear program, the input of the algorithm changes as P and m have to be additionally considered, and the objective function (3.24) must be replaced with (3.28):

$$\text{maximize} \quad \sum_{i=1}^{n} \sum_{j=1}^{r} \sum_{s=1}^{m} P_{is} j x_{ijs}. \tag{3.28}$$

Linear Extrapolation

For this method, we assume that the head movement of viewers follows a linear trajectory over a short period of time. We extrapolate the head rotations of the three head rotation angles θ, ψ, ϕ, i.e., yaw, pitch and roll. A prediction horizon of $p \in \{2, 3...10\}$ seconds is used to estimate the viewport during the next p seconds. While a short prediction window leads to more accurate results, a prediction window smaller than 2 s is not recommended since the video must be split into shorter segments which leads to worse encoding efficiency according to [138]. Therefore, we only use $p \geq 2$ for comparison with other methods.

However, if we base the download purely on live predictions, tiles must be downloaded very late, and the buffer cannot be filled for more than the prediction window p. This is expressed through the point in time D_i^1 at which the prediction is done. In Figure 3.20, we see that for any prediction window p and segment length τ it is $D_i^1 = D_i^2 + \tau - p$. We therefore have to replace the viewport Y with the predicted viewport Y^* and we add the point in time D_i^1 where the

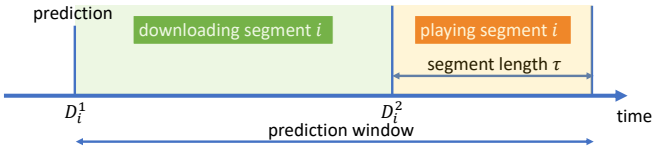


Figure 3.20: Sketch of the deadlines D_i^1, D_i^2 and prediction window in linear extrapolation and machine learning prediction.

prediction occurs as the earliest possible download time. Constraints (3.27) must be replaced with the following constraint (3.29).

$$\sum_{i=l}^{k} \sum_{j=1}^{r} \sum_{s \in Y_i^*} S_{ijs} x_{ijs} \leq V(D_i^1, D_k^2) \qquad (3.29)$$
$$\forall l \in \{1, \ldots, n\}, \forall k \in \{l, \ldots, n\}.$$

Machine Learning Prediction

In a next step, we assume that the head movement patterns of users are similar for the same video and not only linear. We therefore change the method to obtain the viewport prediction and machine learning instead of linear extrapolation. As the resulting prediction is the same, we can still rely on the same linear program that was used for linear extrapolation described in Section 3.6.1.

To obtain a non-linear viewport prediction, we will rely on machine learning to learn head movement patterns from previous viewers and to predict them for the next viewer. For this, we use a neural network that consists of a 4×1000 input layer, three hidden RNN layers with 256 neurons each and a 4×1000 output layer. The RNN uses a softsign activation function. We experimented with many different configurations for the RNN and used these since they returned the best results. The input consists of the four parts q_1, q_2, q_3, q_4 that define the quarternion that describes the head rotation as described in [149]. To speed up

the RNN, we quantize the head movement traces of the users to the 1000 nearest values each.

During training, in each epoch we randomize the order in which the training users are selected. Training is conducted for 2000 epochs for each combination of parameters. We tested a prediction horizon of $p \in \{2, 3, ...10\}$ seconds to estimate the viewport during the next p seconds. Due to the same encoding efficiency problem described in Section 3.6.1, we sue $p \geq 2$ for comparison with other methods.

The RNN returns a quarternion for each time step. From it we determine the viewport Y_i that the user is expected to view during segment i. We also experimented with a CNN with an LRU, but it performed worse than the RNN. Thus, we do not investigate this approach further in this section.

3.6.2 Performance Evaluation of Viewport Prediction

This section describes the methodology and discusses the results of our performance evaluation of the three different viewport prediction methods.

Performance Evaluation Methodology

For the evaluation, 51 head movement traces from [148] are used for the viewport prediction. The head movement traces start 40 s after the start of the video and are 70 s long. For the evaluation we only consider these parts of the video. To ensure that training and testing data is not correlated in the neural network, we use the head movement traces from 40 users exclusively for training and eleven users for testing. For the sake of comparability, we only used these eleven head movement traces for all evaluations.

We implemented the optimization program in Gurobi in Matlab. For the value function w_{ijs}, we chose the resolution j. This means, we focus on maximizing the resolution. To create a realistic streaming experience, we used the eight throughput traces that were recorded according to [169], concatenated them and used consecutive intervals of 100 seconds. This resulted in eleven samples, one

sample for each user of the testing group, that is evaluated. We scaled down the throughput with a factor of 0.05 so that a realistic adaptation scenario is obtained.

For our experiments, we downloaded the YouTube video *2OzlksZBTiA* in eight resolutions: 144p, 240p, 360p, 480p, 720p, 1080p, 1440p, and 2160p. Using ffmpeg, we determined the segment size in Bytes and the segment duration of 5.33 s for each segment in each quality. We divided each segment into three subsegments with a length of 1.78 s. This allows for shorter deadlines to download the segments, which is necessary for a short prediction horizon in the RNN and the extrapolation. Furthermore, we partition each segment into 64 tiles (8x8) with equal bit rate to allow for spatial adaptation.

In the following, performance evaluation results are given considering this video, the head movement traces, and the bandwidth traces as presented above. A one-to-one mapping was used for the traces, i.e., the same bandwidth trace was always used with the same head movement trace.

Comparison of Viewport Prediction Approaches

Figure 3.21 gives an overview of how the five methods perform, that we compare in this study. For each user, the mean resolution of the tiles in the viewport is calculated for each approach. At the y-axis, the mean for all users is presented while we use 95 % confidence intervals. The same confidence intervals are used all other figures. On the x-axis you can see the prediction window. The highest reachable resolution lies between 1080p and 1440p on average, shown by the blue line as baseline. The current YouTube approach (green) of naively downloading the whole video in the same quality can only reach an average resolution of slightly above 480p on average. If we determine a probabilistic viewport using the viewing behavior of other users (red), we can reach a resolution between 720p and 1080p on average, which is in the middle between the optimal and the current approach. The independence of the prediction window is a large advantage of this approach since it allows a large buffer. This helps to avoid stalling and keep a steady resolution if the network is unstable.

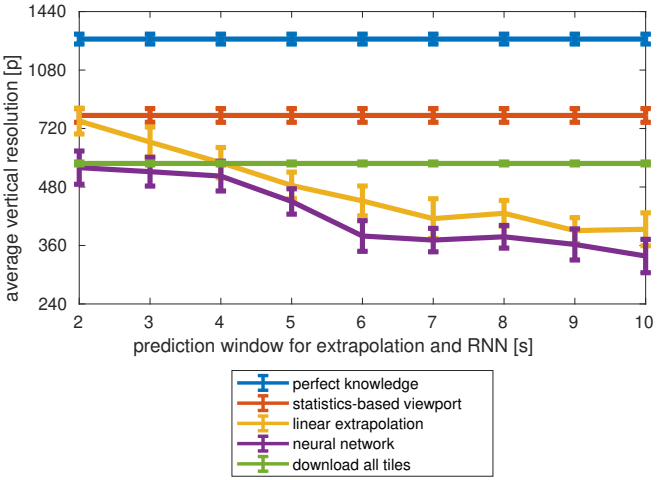
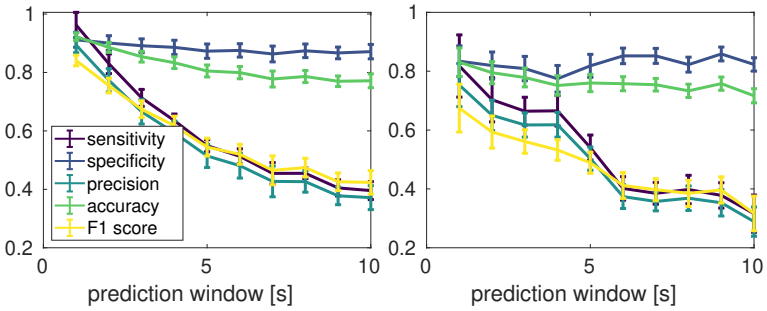


Figure 3.21: Impact of the prediction window size on mean quality over all users. Blue, red and green curves do not depend on prediction and are only given for comparison.

The neural network approach performs worst in this case. For a prediction window up to 4 s, the confidence intervals overlap with the current YouTube performance (green). Thus, no statistically significant difference can be detected. For larger prediction windows the performance becomes even worse. Comparing the linear extrapolation approach with the neural network approach, for a prediction window of 2 s, the linear extrapolation performs better. However, even in this best case, it is only on a par with the probabilistic viewport approach. But since the segment size must be smaller than the prediction window, this can lead to a lot of overhead as concluded in [139]. For larger prediction windows also the performance drops, and no statistically significant difference between the linear extrapolation approach and the neural network approach can be seen based on the 95 % confidence interval.

To sum up, we conclude that the probabilistic viewport prediction performs best if no perfect knowledge is available. Only for a small prediction window of 2 s, the linear extrapolation is on a par with the probabilistic viewport approach, but for larger prediction windows the performance of linear extrapolation and neural network decrease even below the naive baseline (green). Since the playback resolution is only one factor for a QoE analysis, in the following we focus on analyzing quality variations during playback resulting in quality changes.

The average viewport approach (red) and downloading all tiles in the same quality (green) show the smallest variance of all methods. The two user-centric prediction methods, extrapolation (yellow) and RNN (purple), have a high variance which indicates many quality changes.



(a) Extrapolation (b) Neural network

Figure 3.22: Mean value of the classification functions for the neural network and the extrapolation. The value of the y-axis is described in the legend.

Live Viewport Prediction Performance

For a more detailed analysis of the live viewport prediction performance, different prediction window sizes are compared in this section. For the RNN and the linear extrapolation, we regard a viewed tile as a positive and tile that is not viewed as a

negative. In Figure 3.22(a) and Figure 3.22(b), we see their classification functions. In both cases, we get a high accuracy and a high specificity, even for larger prediction windows, because during each segment, only a small subset of all tiles is viewed. However, the sensitivity is very low for large prediction windows. This means that the viewed tiles are not correctly identified. Therefore, we also receive a low F1 score. The fact that our RNN is worse than the extrapolation, indicates that this method should not be applied with such a small data set.

Discussion

From the results of our performance evaluation, it can be seen that the probabilistic viewport approach is the most successful. On average, the quality of the video increases by almost one layer while the mean variance of the quality stays the same. However, in the optimal case, we could still improve the average quality by more than one quality layer. Nevertheless, the probabilistic approach has the great advantage that it is not time constrained. This means that we can have a large video buffer, whereas user-centric live prediction only allows for a small buffer since we can only predict a few seconds in advance. A large buffer helps to reduce the frequency of stalling and quality switches. A disadvantage of this approach is that we need to acquire the viewport of many users, in order to apply it. A large enough group of malevolent users may sabotage a video's quality by viewing 'uninteresting' tiles so that other viewers would only download these tiles in high quality.

The extrapolation approach with a prediction window of up to two or three seconds performed better than the current naive approach of YouTube to download all tiles in the same quality. However, it has the disadvantage that the buffer is very low, and a short segment size is required. On the positive side, no additional user information is required. This makes it the preferred method for 360° live streaming scenarios. While the RNN did not reach good results, we believe that there is potential for applying neural networks in viewport prediction and further research with more samples may allow for better results.

3.7 Lessons Learned

While there exist many adaptation algorithms, each is optimized for a different set of scenarios and metrics. However, optimal values can never be reached for every situation. It would be useful to know what potential for improvement remains, given an algorithm and a scenario.

The goal of this chapter is to find out how adaptation can be conducted optimally in HAS. To achieve this, we use a collection of linear programs that assume perfect knowledge about the future throughout to optimize the video quality and the number of resolution switches while avoiding stalling events.

As a first step, we use a queueing model for adaptive video streaming to get an understanding on how key performance indicators are impacted by configuration at the buffer or at the video content provider. Our analysis shows that at least five quality layers should be offered, to avoid most stalling.

Additionally, we use two methods to quantify the decrease in efficiency when downloading the same segment multiple times in different quality levels. The first method is a fast heuristic approach based on historical data. The second method is based on an optimization problem formulation. Our results show that this method is inefficient and that on average 20% of the videos could have been downloaded in a higher quality. Furthermore, our optimization program shows that stalling can be prevented in 94% of these cases. At the same time the initial delay can be kept below $10\,\mathrm{s}$ in 95% of cases.

We investigate the trade-off between the average quality and the number of quality switches in a video streaming session. For this purpose, we use a quadratic program that includes these two values in its optimization function in order to receive optimal values. Our results show that high average quality can be achieved with few switches while a very high number of switches is necessary to achieve the highest possible quality. We conclude that it is advisable to rely on conservative strategies that do not switch with a high frequency. However, it is still subject of future work to what degree the number of switches actually impacts the QoE.

Next, we conduct a measurement study in which we observe that the QUIC protocol is not always fair when multiple users share the same bottleneck in video streaming. We investigate three approaches to fairness and extend above quadratic program to include fairness. We find that the fairness of a video streaming system with multiple users can be improved significantly at little cost in terms of quality or switches.

Finally, we venture into 360° videos and investigate three different approaches for viewport prediction. For each of these approaches we propose a linear program that takes the prediction as input and uses it to optimize the video quality. Our reslts show that a statistics-based adaptation is recommended when user viewports are available, since this approach leads to the highest average quality and the lowest variance. If live prediction is required, e.g., in live streaming where no historical statistics about the users' viewports are available, linear extrapolation performed best although it only worked well for a prediction window of two seconds.

4 Resource Efficiency Measures in Mobile Video Streaming

According to a Conviva report [170], the QoS in video streaming has improved significantly between 2017 and 2019 due to increased bandwidth capacities. In particular, the rebuffering ratio has decreased from 1.0% to 0.5% and the average video bit rate in which videos are viewed has increased from 3.3 Mbit/s to 4.6 Mbit/s. While the throughput of mobile networks has increased drastically during the past decades, the energy density of lithium-ion cells has only quadrupled since they became commercially available in 1991 and may soon hit a limit [171]. The average monthly data volume per mobile Internet subscription in Germany was 850 MB in 2017[1]. In comparison, one hour of HD video (720p) uses about 900 MB. So, even though energy and data capacity are important limits in mobile video, currently, video service platforms do not use energy and data efficient adaptive streaming mechanism. In academic research, a first significant contribution was made in [172] where a data conserving algorithm is presented for non-adaptive streaming.

In Table 4.1, the challenges and interests of different stakeholders are summarized. The mobile user is most interested in high QoE, while he needs to manage his limited resources. The ISP is interested in maintaining an infrastructure that allocates resources efficiently with high QoS. The video service provider wants its users to be highly engaged with its platform. Moreover, he aims to reach as many users as possible, including mobile users. The main goal of this chapter, is to find

[1] https://www.statista.com/statistics/469121/mobile-internet-monthly-data-volume-per-user-germany/

stakeholder	challenges	interests
mobile user	limited resources	high QoE
ISP	efficient resource allocation	high QoS
video service provider	mobile access	high User Engagement

Figure 4.1: Stakeholders of mobile video streaming.

solutions that improve the resource efficiency of the user while maintaining high QoE and not interfering with the other stakeholders in a negative fashion.

With the steady increase of the total Internet traffic [173] caches seem like a good option to reduce it by bringing the content closer to the end user. Many websites use CDNs to get their content closer to the consumer. Caching data inline on the path between the origin content source and the end users has been another solution for many similar problems in the past. Here, we focus on a caching approaches situated at home-routers, wherein content is placed particularly close to the edge. We therefore conduct a field study at public WiFi hotspots where we install caches to investigate the potential caching could have to save Internet traffic in a practical experiment. We discuss challenges that HTTPS introduces to caching and analyze the performance of the caches in terms of scalability. Our main research question is: *To what degree can caching be used for video streaming and how well could it perform?*

In order, to improve the energy consumption and data efficiency of mobile videos, we must understand how such videos are consumed. Therefore, as a first step, we want to identify the behavior of YoMoApp users who view videos in a natural setting on mobile devices. For this, we use a provided data set from a four-year long user study with a mobile phone application that allows to watch YouTube videos. This data set was partly published in [174]. We analyze the frequency and the correlation between different user actions. Particularly interesting parameters that we investigate include stalling, pausing, seeking, quality changes, and device rotations. Our research question is: *How do mobile video streaming users behave and interact with the video?*

As a third step, we present a new adaptation algorithm for mobile HAS that is based on KLUDCP [96]. The proposed algorithm uses audience retention statistics of the video that is currently watched to keep the buffer low during periods where many users abandon the video. This way, fewer buffered video segments are discarded and thus less traffic is wasted. Otherwise, the algorithm downloads video segments in an on/off pattern based on an algorithm presented in [172]. This way, the device must not always be fully connected with the cellular network, leading to lower power consumption. In practice this algorithm is easy to implement for video streaming providers since they have access to these statistics. We compare our algorithm with its baseline (KLUDCP) in terms of application layer QoS (i.e. number of stalling events, average quality, and number of quality switches) and in terms of energy savings and data savings. For this purpose, we conduct a simulation which uses viewing statistics[2] from real YouTube videos, real mobile bandwidth traces (WiFi, 3G [175] and LTE [169]) and appropriate energy models. In the simulation, users watch videos and may skip ahead in a video or abandon it based on the audience retention statistic of each video. This is the first time that real user behavior is used in an adaptation algorithm and in its evaluation.

This chapter is based on [11, 16, 19] and its remainder is structured as follows. In the next section, we discuss background and related work on user engagement, energy efficiency, and data efficiency. In Section 4.2, we discuss our field study with public WiFi-caches and to what degree caching could be used to reduce traffic. Section 4.3 describes our study on the behavior of mobile video users. In the next section, we present a novel adaptation algorithm that is very energy- and data-efficient. Finally, we sum up the lessons we learn in this chapter.

[2] https://support.google.com/youtube/answer/1715160

Table 4.1: Six levels of video streaming from a user centric point of view.

level / layer	parameter group	example parameters
network	QoS	packet loss, bandwidth, latency
application	QoS	stalling, resolution, initial delay
user	QoE	MOS, SOS
single video	user behavior	viewing duration, interactions
video session	user behavior	number of consecutive videos
user history	user behavior	return rate, average session length

4.1 Background and Related Work on User Engagement and Resource Efficiency in HAS

4.1.1 User Engagement in Video Streaming

When analyzing video streaming from a user-centric point of view, there are different angles from which video streaming and its users can be investigated that differ in scale and layer, compare Table 4.1.1.

The User Engagement describes the time or percentage of the video viewed, i.e. the time between starting to play the video and reaching the end of the video or abandoning it subtracted by pauses or stalls. In [176], a large-scale study of the User Engagement of 1.5 million unique users is conducted over a seven month period. They focus on user arrival rates and request patterns and investigate the impact of network and application layer QoS on User Engagement. The authors find that the buffering ratio has the highest impact on User Engagement. The authors of [177] investigate the impact of different application layer QoS metrics on User Engagement for different types of content. They use a data set with over 1 million users. Based on their results, it is shown that there exists a strong correlation between QoE and User Engagement. In [25], they measure QoE-metrics and User Engagement from various sites, different types of content (short VOD, long VOD, and live video), and also distinguish other kinds of parameters. Their results show that a high buffering ratio lowers User Engagement, with the

impact being stronger for short videos. Similarly, a high bit rate has a significant impact in the live scenario while it does not in VOD. Other metrics that are used to indicate User Engagement include the number of comments, the number of votes or likes and the like-dislike ratio. These metrics are usually applied on a video level or on a content provider level [178]. In [68] traffic during a single live event is measured and the impact of QoE metrics on User Engagement is analyzed. Their results show that the buffering ratio and the bit rate have a high impact on User Engagement. Further, they noted that the video play time may depend on various other factors such as user behavior. A correlation between QoE and User Engagement was also recognized. A 2014 paper [69] conducted a large scale measurement study that looked at the abandonment rate — which can be another appropriate User Engagement metric — for mobile video streaming. Using data from the study, a model is proposed that can predict User Engagement in mobile video streaming with a high accuracy based on network statistics. In two further publications, Balachandran et al. [70, 71] measured User Engagement and video session quality and run machine learning algorithms on it. Through this effort, they highlight the challenges of obtaining a robust video QoE model from such metrics. A related work from Krishnan et al. [74] puts viewer behavior in relation to video quality metrics. Of note is the observation that an increase in the initial delay of a video stream also directly leads to a higher abandonment rate. Video content related reasons for declining User Engagement and abandonment are described in [179] at the example of a lecture video. The impact of application layer QoS on the User Engagement is also researched in [180]. The authors define new metrics to characterize the User Engagement for a data set of over 1000 non-mobile YouTube viewers. Their results show that a negative video bit rate change leads to more abandonment than stalling and that even positive bit rate changes lead to abandonment. Other reasons for abandonment have been studied in a field study in [116] using the browser plugin YouSlow. The authors of [181] propose a comprehensive QoE model that includes the user state and the user behavior. They analyze QoE, the user state and the user behavior from the perspective of the user, the system and the service provider. Furthermore,

they discuss how the price and the energy consumption of a service may impact QoE and user behavior.

While a video is playing, users have various options to control the playout of the video. Users can skip ahead to a later point in the video. This is also called seeking or fast forwarding. If they seek a video segment that is not yet downloaded, stalling will occur. However, since this is an expected event, it is not as annoying as other stalling events but more like initial delay [182]. Users can pause and resume videos. Usually, the video download continues while the video is paused until the buffer is filled. Thus, users often use this option to avoid stalling events when they have a low throughput. Similar to the YouTube App, in the YomoApp full screen mode can be activated by either pressing the respective button or by rotating the device into a horizontal position (landscape mode). Normal mode can be re-entered by reversing this action. Furthermore, users have the option to manually change the video resolution, the volume and the speed at which the video frames are replayed. The typical video behavior of about 300k users is investigated in [183]. The authors present a model for the user arrival process and for the stream control. They find that there exist typical sets of viewer actions such as fast forward, followed by pause and rewind.

Frequently, users will watch another video after finishing or abandoning a video. We define the process of starting the YomoApp, watching one or more videos, and closing the YomoApp as a video session. If a viewer watches two or more episodes of a series in one session, it is called binge-watching [184]. In this chapter, we investigate the number of videos that a video session contains. Other interesting metrics include the session duration and the time delay between end of video and start of next video. If we consider multiple sessions of the same user, we look at the user's history and consider their behavior over long time periods. Particularly interesting parameters include the rate with which the user returns to the service, the average session length, the start and end date of use or subscription to the service and the money or resources spent on the service. These parameters are particularly interesting for strategical, financial

and marketing decision making. However, they are out of scope of this work and will not be discussed in greater detail.

4.1.2 Energy and Data Efficiency in Mobile Video Streaming

Energy and traffic are often limited resources in mobile scenarios. Let us first investigate the energy consumption of mobile devices. During a video stream, the greatest share of the energy is typically consumed by the display of the device. The amount of consumed energy mainly depends on the display type, the display's brightness, and a frame's colour composition [185]. Modern devices already adapt the brightness depending on the surrounding brightness and on the battery status. The decoding of the downloaded video also consumes energy. However, this is usually done by specialized hardware components which do not consume a significant amount of energy during the decoding [185]. In [185], a energy model for the power consumption of video decoders is discussed. It can be seen that some energy can be saved by playing a video in low quality.

The Radio Resource Control (RRC) is a protocol used in UMTS and LTE. Its functionality includes the management of resources and inactivity timers which significantly impact the power consumption of a device. Depending on the (cellular) network technology used, there are different states of connectivity that a device can assume. In UMTS, the most important RRC states include DCH, FACH, PCH, and IDLE [186]. Their energy consumption is discussed in detail in [187]. DCH is the state which provides a dedicated channel to the user allowing for high throughput but also consuming the most power. FACH is used by applications with very low throughput and consumes about half the power compared to DCH. While in the PCH the user device cannot receive or send packets. However, it can receive notification whether there are downlink packets and then change its state to DCH or FACH. This state only consumes about 1-2 percent of the power compared to DCH. In the IDLE state, the device is disconnected and the energy consumption is similar to PCH. If the device is not

in IDLE and there is no traffic, it will change its state to a less energy consuming one after a few seconds. The exact duration is defined by inactivity timers. In LTE, there are two RRC states, CONNECTED and IDLE [188]. If we try to compare them to UMTS states, CONNECTED is similar to DCH and IDLE is similar to PCH. Furthermore, LTE uses DRX to continuously listen for data with reduced energy consumption. A very detailed overview of DRX can be found in [189].

Qian et al. [190] investigate different RRC inactivity timers in video streaming over 3G cellular networks. They discover a performance inefficiency due to tail effects and state promotion overhead. They find that each application has its optimal value for the inactivity timer. In [191], they present a framework that optimizes tail times resulting in a significant reduction of energy consumption for various Internet applications. Hoqoe et al. [192] discover that video streaming platforms use different streaming techniques for different devices, players, and video qualities. They discover that there is room for optimization in terms of energy efficiency for every technique.

Seufert et al. [193] use throughput traces of 2G, 3G, 4G and WiFi networks to investigate how effective it is to offload mobile traffic to WiFi hotspots. They find that the low throughput of WiFi networks leads to lower QoE and higher energy consumption compared with 4G. WiFi offloading is recommended if only 2G or 3G networks are available but not for 4G. In this chapter, we conduct a similar simulation, which utilizes 3G and LTE network traces to simulate HAS in a mobile environment. Further, we simulate user behavior and evaluate the energy consumption of different adaptation strategies and different devices to evaluate HAS with similar network traces and different tail times.

Schwartz et. al. [194] evaluated four different video streaming mechanisms, in respect to QoE, energy consumption, User Equipment, and wasted traffic. They show, that their streaming mechanism, which uses a buffer with two thresholds, offers the best trade-off between energy consumption and wasted traffic. They use basic probability distributions to model the user behavior while we use real user statistics in our evaluation. Furthermore, their work does not discuss

adaptive streaming mechanisms since it was written in a time before adaptive streaming was popularized.

Siekkinen et al. [195] measured 20% energy savings in mobile video streaming when shaping the traffic received from 3G and LTE networks into traffic bursts. Energy could not be reduced further due to YouTube's background traffic, which was interfering with the traffic shaping and causes unexpected transitions of the RRC. Traffic shaping also provides a good balance between saved energy and signaling overhead. In [172], they developed a scheduling algorithm, which relies on viewing statistics to reduce the energy consumption and traffic overhead in mobile video streaming. They defined a scenario, where users can abandon the playback at any time and developed an algorithm, which predicts the user behavior based on the viewing statistics. Depending on the wireless interface that is used, the algorithm calculates the energy and traffic optimal download schedule. We adapt their scheduling algorithm to develop a HAS adaptation strategy, which optimizes the quality, energy consumption, and traffic waste. Furthermore, we follow a similar idea and utilize audience retention statistics to perform a user centric traffic shaping. However, our strategy is built around HAS since it is very common in video streaming applications. In addition, we have more complex video sessions, where multiple videos are watched, and video content may be skipped.

4.2 Viability of Caching in an Era of HTTPS

Popular Internet videos are transmitted very frequently and may only be interesting in a small geographical region, e.g., due to the used language. Therefore, caching popular videos at the edge seems like a resourceful option to save traffic at first glance. However, most traffic in the Internet is encrypted, in particular videos. It is very likely that YouTube will continue delivering its content via HTTPS and using its Content Distribution Network (CDN) for better content propagation. The consequence is that content will be difficult to cache at a forward or inline cache, yet CDN servers are typically already situated at the

network edge which leads to diminishing returns for additional caches. This inspired us to investigate the general feasibility of edge caching with the increasing deployment of HTTPS in the Internet, in particular. Furthermore, we are interested in the performance of caches regarding their loading time and the number of simultaneous connections since little research is done in this regard. Our research questions can thus be formulated as follows: *What share of Internet traffic can be cached at WiFi hotspots? How well does a cache perform under high load?*

4.2.1 Measurement Study

This section discusses the methodology of our experiments which can be divided into two parts. First, we set up a prototype of our caching system in public locations to investigate its behavior in a realistic environment. Second, we benchmarked the system to evaluate its performance and compare it to the proposed queueing model.

Setup

First, let's introduce our setup. For the practical evaluation of edge-caching we built a low-cost setup that can be installed together with standard access points. We used standard consumer access points (`TP-Link WDR-4300`) for the WiFi and a Banana Pi, a credit-card sized computer (`ARM Cortex A7`, 2 GB RAM), with an SD-card as caching storage. This compact form-factor with its low-power cost properties allowed for the placement in locations, where no large-scale cache could be employed, e.g. directly at a WiFi access points. All devices are then connected via 1 Gbit/s-Ethernet using the access point's on-board switch.

The general setup works as follows: The access point announces the public WiFi network. All HTTP-traffic is then routed to the cache, acting as a regular transparent proxy. All other traffic is directly routed to the Internet without the intermediate cache. Due to legal reasons, the traffic was routed through a secondary TP-Link router over a VPN tunnel.

The access point is running OpenWRT[3], a Linux-based distribution for access points, that allowed us to route the traffic using `iptables`. Additionally, we were able analyze the traffic with `iptraf`[4] in order to determine the ratio of cacheable HTTP traffic compared with the overall traffic. The cache runs Linux and the proxy server `Squid`[5]. We chose `Squid` over other proxy software solutions, as it is actively developed and its cache replacement strategies as well as the cache size are configurable. This gives us the possibility to extend our research on this basis.

Real-world Test

We chose two locations for evaluation: a computer club, where a WiFi access point existed before, and an apartment, with a bar within its radio coverage area. The caching proxy logged all HTTP requests and whether they invoked a hit or a miss. The experiments were conducted in August and September of 2016. In the first location — the computer club — the experiment ran for five days, and nine days in the second location. In the computer club, a VDSL Internet connection with a downlink speed of $100\,\mathrm{Mbit/s}$ was available, while at the other location a Cable connection with $50\,\mathrm{Mbit/s}$ was used.

Benchmarking Study

For the purpose of benchmarking, we tested the cache in a controlled environment using wired $1\,\mathrm{Gbit/s}$ connectivity for all devices. First, we determined the page load time of the nine most popular websites in Germany in 2016 according to Alexa [196], that did not use HTTPS by default with an empty cache, so that a cache miss is guaranteed to occur. Second, we repeated the experiments with a cache, where the web page is already stored (i.e., a cache hit). For comparison, we also measured the page load time without any intermediate cache. This experiment is indented to show the potential of caching. The loading time of

[3] https://openwrt.org/
[4] http://iptraf.seul.org/
[5] http://www.squid-cache.org/

pages was determined using `Firefox 42.0` with a fresh browser profile and an empty cache. The add-on `app.telemetry`[6] measured the page load times. `App.telemetry` uses the `onLoad()`-event to determine when a page is fully loaded. During our tests, this reflected the perceived loading time. If a request was not completed within $\theta = 30\,\mathrm{s}$, the request expired. We also observed, that the displayed advertisements change every time the web pages were loaded, while the page's actual content remained the same. To obtain comparable results, we therefore used an ad blocker (`Adblock Plus`[7] with the filtersets *Easylist Germany+Easylist*). This measure made it possible to directly compare the page load time of multiple requests of the same page.

To investigate the behavior of the cache under load, we used the tool `Siege`[8] to generate requests for a file with a size of 100 kB from another web server. At the same time, we measured the loading time of a web page like in the previous experiment. Different levels of load were generated once again with `Siege`'s capability to simulate more than one concurrent simultaneous user requesting the specified file. We tested our setup with $0 - 700$ users, which do not wait for the result of their request. The inter-arrival time of the requests follows a continuous uniform distribution $\mathcal{U}(0\,\mathrm{s}, 1\,\mathrm{s})$, hence the arrival rate of $\lambda = 1/\mathrm{s}$. Each of the individual components ran on a separate network node, i.e. a request generator, the cache, as well as the web server for the test file.

Note, that no steps were performed to ensure absolutely realistic load scenarios. For example, `Squid` uses in-memory caching, which cannot be deactivated. Due to this, the test scenarios would probably result in an unrealistically good performance. Additionally, the `Siege` load tests do not produce realistic load patterns, they are only meant to generate a steady load baseline at the cache.

[6] http://www.apptelemetry.com/
[7] https://adblockplus.org
[8] https://www.joedog.org/siege-home/

ratio	club	bar	Google 08/2016 [198]	Google 05/2020 [198]
HTTP	6.6 %	16.6 %	15.0 %	5.0 %
HTTPS	93.4 %	83.4 %	85.0 %	95.0 %

Figure 4.2: Share of HTTP and HTTPS traffic.

Cacheability of Requests

Next, we examine the share of cacheable requests, which is identical to the ratio of secured to unsecured transmissions. The results are compared to encrypted traffic across Google in Table 4.2. Both data sets show similar characteristics: The share of HTTPS traffic is the largest with over 80 % in the bar and over 90 % in the computer club. The second rank is taken by HTTP traffic. Other traffic types make out less than 0.1 % of the total traffic. 45 % of page loads use HTTPS [197] and many of those consist of large files such as YouTube videos which leads to a very high share of HTTPS traffic.

The total hit rate of our cache only lies at 12.6 %, resulting in an overall reduction in non-encrypted traffic of 1.6 %. Interestingly, the highest caching efficiency was achieved in RSS traffic and ingress traffic was significantly reduced (by 58 % at the computer club). However, RSS does not amount to a large share of total traffic.

4.2.2 Cache Performance

In order to investigate the CPU performance of our cache, we now present our system model and compare its results to the measurements.

System Model

We model the cache as a $M/M/1/S$ loss system with an average arrival rate λ, average service rate μ and a finite queue with length S. In practice a request

is blocked if its waiting time expires a threshold θ. We consider this expiration threshold by setting the queue length S to $S = \theta \cdot \mu$.

The system dynamics can be described by a birth-death process with states $\{0, 1, \ldots, S+1\}$ and transition rates λ and μ. Using the completeness relation, the steady state probability $x(i)$ that i jobs are in the system on arrival of a request is calculated by

$$x(i) = \frac{\left(\frac{\lambda}{\mu}\right)^i}{\sum_{j=0}^{S+1}\left(\frac{\lambda}{\mu}\right)^j} .$$

The blocking probability p_b is defined as the probability that the system contains $S+1$ jobs upon an arrival. Then the server and each slot in the queue is occupied.

$$p_b = x(S+1) = \frac{\left(\frac{\lambda}{\mu}\right)^{S+1}}{\sum_{j=0}^{S+1}\left(\frac{\lambda}{\mu}\right)^j} . \tag{4.1}$$

As long as the system is serving any requests the CPU load ρ^* can be estimated as estimated as

$$\rho^* = 1 - x(0) = 1 - \frac{1}{\sum_{j=0}^{S+1}\left(\frac{\lambda}{\mu}\right)^j} . \tag{4.2}$$

Results

For the analysis of our benchmark results we use the page load time and the CPU load as performance metric. Figure 4.3 depicts results for popular (according to [196]) German web sites. For some of these pages the use of a cache does not lead to a statistically significantly different load time (`amazon.de`, `ebay.de`, `gmx.de`, `spiegel.de`, `chip.de`). For others, we note that a cache miss leads to a load time of more than 2 s which is considered unacceptable for web

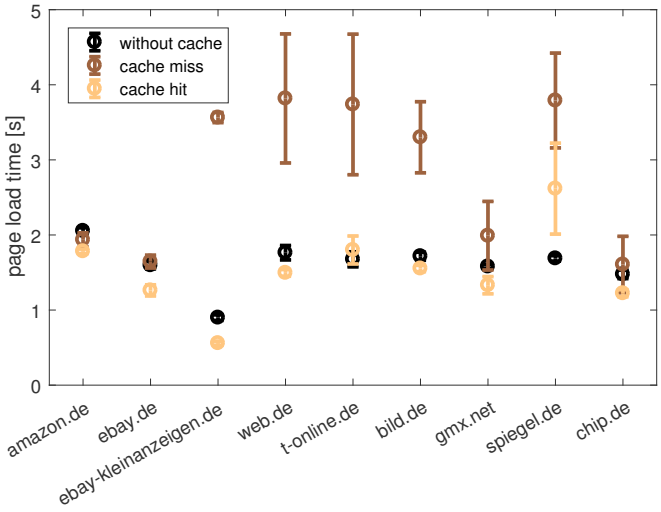


Figure 4.3: Comparison of loading times of popular websites with and without cache.

pages [199]. While a cache hit will always improve the load times compared with a cache miss, there are some scenarios (e.g. `spiegel.de`) where even the presence of a cache can negatively impact the performance and result in an overall increase of the load time no matter if it's a cache hit or miss.

Next, we want to determine how the page load time increases if the CPU is overloaded. Results are shown in Figure 4.5. An individual request of the page `web.de` was measured to have an average page load time of 4 s (variations can be attributed to the dynamics of the page's contents). When the cache is subjected to an increased CPU load by saturating it with requests, the page load time in seconds increases linearly based on the mean arrival rate of request λ according to the function

$$f(x) = 0.0198\,\lambda + 4.568\,.$$

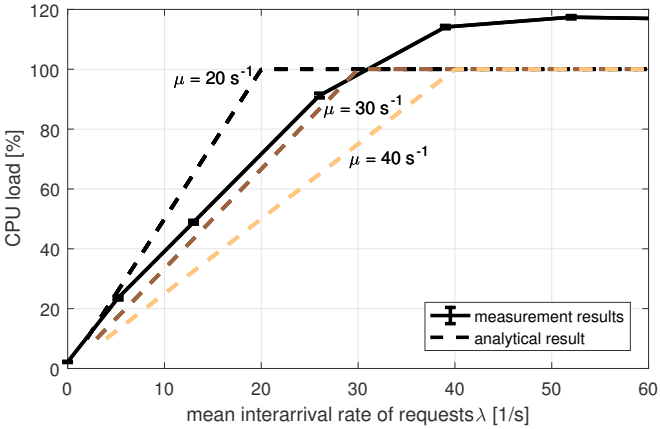


Figure 4.4: Comparison of measurement results and analytic results for the CPU load of the cache based on the mean arrival rate λ and the mean service rate μ (in seconds). Analytical results are based on Equation 4.2.

In Figure 4.4 we investigate the number of requests that the cache can serve simultaneously. This is done by observing the CPU load and the duration of the longest transaction. We notice that up to 30 simultaneous connections do not impact the performance of the cache. If there are more than 40 requests per second, the cache is overloaded and it takes several seconds to process a request. Since we used a dual-core processor for the experiments, the CPU load may exceed 100 %. The analytical results for the request time-out probability show that it is easy to dimension a cache based on the number of incoming requests, cf. Figure 4.6.

4.2.3 Discussion

The results collected here seem to be in line with our expectations. At both the computer club and the bar a very unique traffic mix could be captured (where

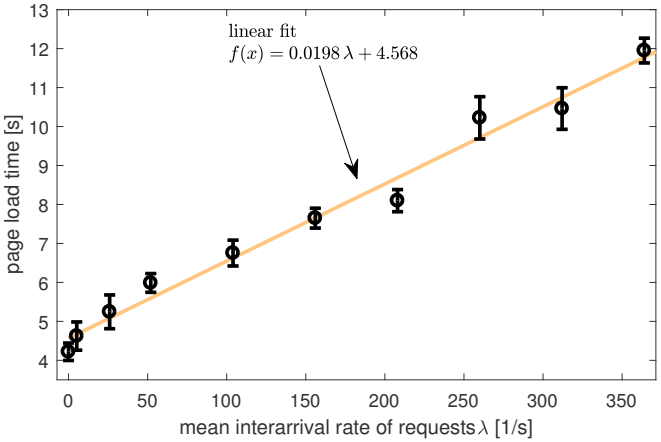


Figure 4.5: Loading time of the web page web.de using a cache that is servicing multiple connections simultaneously with a mean arrival rate λ.

the caches and access points were situated). The specific audience at these locations was quite indicative of a broader future, where almost all connections could be secured. Such a situation may come to pass rather sooner than later, as all involved parties, including browser vendors and Website operators, are increasingly pushing this, e.g. by notifying the users about unsecured Web pages in their browser, or by discouraging and blocking mixed content sites.

In the end this will lead to an inability to conduct any kind of forward and inline caching or operate any other kind of application layer middlebox. The only viable form of caching in this scenario will be reverse caches inside the domain of control of the content provider, with the HTTPS connections' endpoint at the caches. This is the scenario that is already present in today's solutions in the form of CDNs.

But even if this traffic mix scenario does not come to fruition, the results have also shown other drawbacks in such a forward caching approach. Cache misses

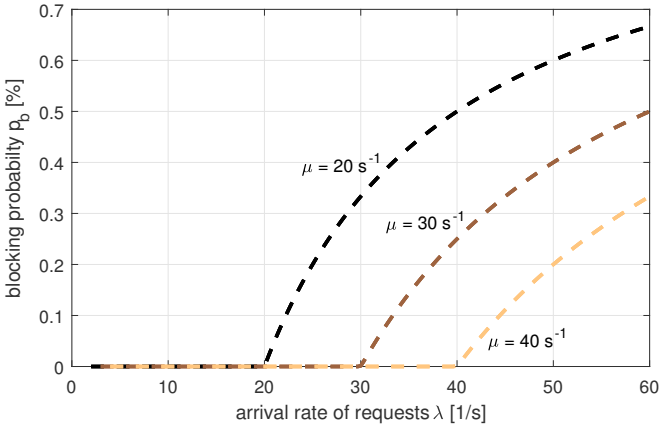


Figure 4.6: Analytical results for the probability that a request times out according to Equation 4.1.

are rather costly in terms of page load time, yet the gains in case of a cache hit might only be marginal. And cache misses will frequently occur due to the low number of target users when the cache is situated so close to the content's destination. Even looking from the perspective of network and service operators caching on the user's premises might not be worthwhile at all, as the amount of bandwidth saved might be minimal, as our experiments have shown.

4.3 Evaluation of User Behavior and Abandonment

Since watching videos in mobile networks is much more challenging than with fixed connections, it is important to understand the user behavior in this scenario very well. In this section, we evaluate the results of a measurement study on the behavior of mobile video users to get an idea on how their behavior might be different from the behavior of desktop users. For the algorithm which we will

present in Section 4.4 the abandonment behavior and the number of consecutive videos are of particular interest.

4.3.1 Data Set and Crowdsourced Measurement

The tool YoMoApp [200] is an application for YouTube crowdsourced QoE measurements publicly available on the Google Play Store. It replicates the native YouTube client, giving it the same functionality as the original app. Personal recommendations and favorites are displayed, and you can play your previously tagged videos. The goal is to provide a measurement methodology to monitor application-related and user based KPIs during a YouTube session. Monitored KPIs were chosen to be main QoE factors of the mobile user. Furthermore, we monitor user interaction such as activating the full screen or changing the playback quality. The app also collects data about the device and the user context, including KPIs such as screen size and orientation, user location, and mobility, and ISP. The existing data set we survey consists of more than 6,000 YouTube video views collected over 70 different mobile operators worldwide and by 451 different users from 2014 to 2018.

4.3.2 Characterization of User Behavior

In this section, we evaluate the data set described in the previous section with focus on user behavior over single videos, multiple videos, and reasons for the behavior. We start by looking at correlations between important measures to point us into the right direction which are summarized in the Appendix in Table B.1. We then investigate the occurrence of stalls and the active behavior of users, e.g., how often they pause videos. Lastly, we analyze the User Engagement and the abandonment behavior of users. The preexisting data set we survey consists of more than 6,000 YouTube video views collected by over 70 different mobile operators worldwide and by 451 different users from 2014 to 2018.

A first surprising discovery was that the User Engagement only has a small negative correlation with the stalling ratio in the YomoApp. This is a contrast

to the YouTube App, for which a stronger impact of the stalling ratio on the User Engagement is presented in [180]. In addition, Section 2.5.1 shows that User Engagement can be directly mapped to stalling ratio in non-mobile scenarios. Furthermore, we found a small positive correlation between User Engagement and the number of quality changes. In [180] a negative correlation between User Engagement and positive and negative switches was shown for non-mobile YouTube users. In addition, we see that the number of seeks has a negative correlation with the User Engagement. We can imagine impatient YoMoApp users to fast forward more frequently as well as abandon videos earlier. The stalling ratio correlates with stalling duration and with number of stalls which is unsurprising since it is a function of it. The number of stalls correlates with the number of pauses and seeks. That makes sense, since stalling occurs when the user skips to a part of the video that is not yet buffered. Furthermore, users that experience stalling like to pause the video or manually decrease quality to load a large section of the video to prevent further stalling events. Mode switches and screen orientation changes are correlated since orientation changes often trigger full screen mode. Usually, a video will go into full screen mode, if the device is rotated. However, on some devices an orientation change does not lead to a mode change, e.g., when the respective option is deactivated by the user. Inversely, when users manually go into full screen mode they usually want to rotate their device to view the video properly. Therefore, both curves are very close together and are strongly correlated (coefficient of correlation of 0.62).

User Interactions

We observed three kinds of events that interrupt the flow of the video: pausing, stalling and seeking. Almost 90% of videos were not paused once and only very few videos are paused more than once. Most pauses are shorter than two seconds. However, several users pause their video for a very long time. More common than pausing is seeking, which occurs in about 75% percent of videos. Stalling was the most frequent event that interrupted the video playout. However, it is the only event that is triggered involuntarily and has a negative impact on the

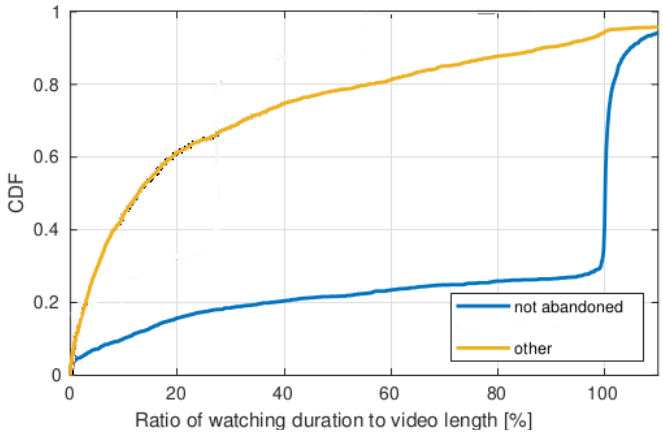


Figure 4.7: Relative User Engagement, i.e., ratio between User Engagement and video length.

user experience. Stalling events were also much shorter than pauses, most of the time. Notice that we do not consider initial delay as stalling.

User Engagement and Video Abandonment Behavior

The above investigated user actions affect and extend the total amount of time a user spends on a video and how long he/she stays on the video page. The time users spend on a video consists of watching, stalling, initial delay, pausing and seeking. For the User Engagement we only consider the time spent watching. In Figure 4.7 we investigate the relative User Engagement as a share of the video length. When a video is not abandoned a time slightly higher 100% is expected. Higher values occur because some people rewind the video to watch certain scenes multiple times. Lower values than 100% in the blue curve indicate that a user skipped ahead in the video. Most YoMoApp users only watch a single video

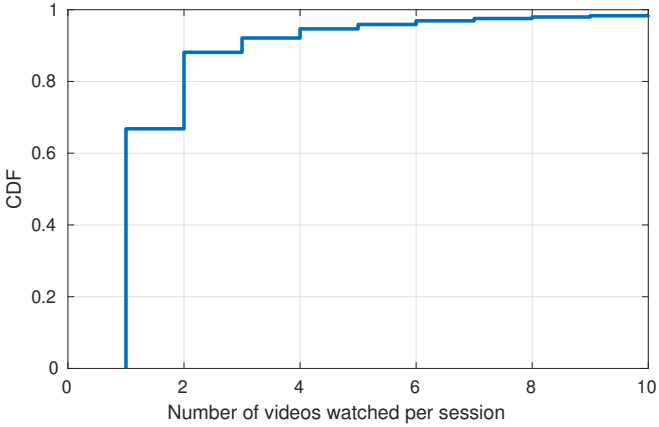


Figure 4.8: Number of videos that were watched in a row by the same user during one session.

during a session, compare Figure 4.8. Only about 1.1% of users watched more than ten videos in a single session.

When a video runs in full screen mode and the user wants to abandon the video, he first needs to leave full screen mode. We suspect that this is a primary reason to leave full screen mode. In Figure 4.9 we therefore determine the time between the last change from full screen mode to normal mode and the point in time where the video is abandoned. Indeed, 50% of users abandon the video within three seconds of leaving full screen mode. However, over 5% of users only abandon the video 50 seconds or later after leaving full screen mode.

Concluding, this data gives us a good insight, that YoMoApp users can behave very differently from each other. Furthermore, we observed typical binge-watching behavior as is common on other video platforms. In the remainder of this chapter, we focus on YouTube users instead of YoMoApp users which is why we use different data on the user behavior for the evaluation in the following section.

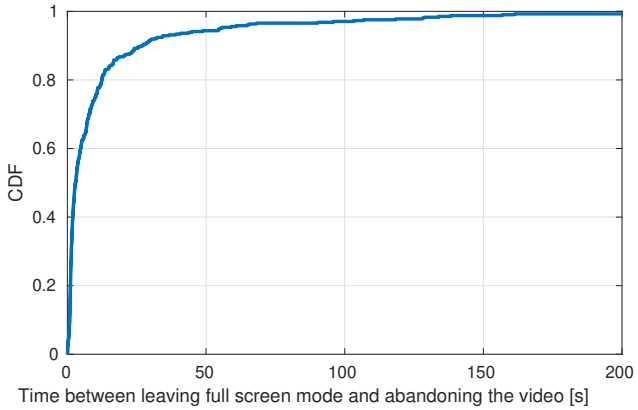


Figure 4.9: CDF of time that passed between leaving full screen mode and abandoning the current video.

4.4 Energy Efficient Adaptive Streaming Algorithm

It has been shown that not only video degradation but also the energy consumption and the battery life while using a service or application has an impact on the user's QoE [201]. We develop a new algorithm for mobile video streaming that aims to reduce the video streams energy and data consumption while delivering high quality. We investigate the performance of the algorithm in WiFi, 3G, and 4G networks in a simulation using using real world-like user behavior patterns. Figure 4.10 gives an overview of the structure of this section.

4.4.1 Adaptation Algorithm

In this section, we present the adaptation algorithm and its components. Our algorithm is a combination of the buffer- and bandwidth-based algorithm from [96] and an energy-efficient schedule for the download of video segments from [202] that we extend for adaptive streaming. We determine the bit rate bw^+ of

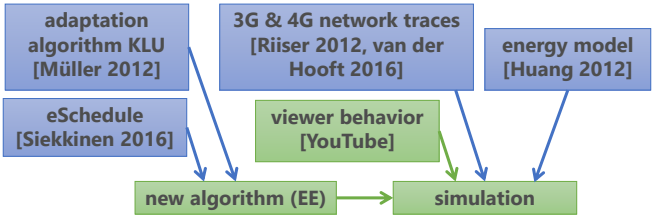


Figure 4.10: Overview of other works that were used by us (blue) and our own contribution (green).

the following segments P_1 to P_2 according to KLUDCP, adding an error variable to avoid overestimating future bandwidth. Since we want to select a quality layer for a batch of segments instead of a single segment, several changes had to be made to the basic algorithm. To avoid stalling when downloading large batches of segments, we estimate the bit rate after the download of each segment. An overview of the used notion is given in Table 4.2.

We define an error $er \in [0, 1]$, which describes the highest percentage, which we expect the bandwidth to sink/rise per downloaded segment. We call this optimistic bandwidth estimation for a rising throughput and pessimistic bandwidth estimation for a shrinking throughput. Equation 4.3 shows the optimistic bandwidth estimation, where the error value rises per segment download and the resulting estimated bandwidth rises per segment download. Equation 4.4 shows the pessimistic bandwidth estimation, where the error value rises per segment download and the resulting estimated bandwidth shrinks. Note, that we hereby extended the model presented in [96] to include an error value and batches of segments instead of a single segment. The buffer level is used to calculate the quality level, so we calculate the expected buffer level after each segment download. The throughput prediction can be replaced by more refined methods, compare [203].

Table 4.2: Notion of variables.

Variable	Definition
P_0	last segment that was downloaded
P_1	first segment of next batch that is downloaded
P_2	last segment of next batch that is downloaded
n	size of batch in segments
bw^+	optimistic bit rate suggestion for next batch
bw^-	pessimistic bit rate suggestion for next batch
tl	tail time
er	predictability of the network throughput
$r_{dl}(P_0)$	throughput during download of last segment
bl^+	optimistic buffer level estimation
bl^-	pessimistic buffer level estimation
$Q(i)$	quality layer in which segment i is downloaded
$br(Q(i))$	encoding rate of quality layer $Q(i)$
t_i	remaining play time of segment i
con	constraints to avoid stalling and guarantee high quality
$p_{abd}(j)$	abandonment prob. at segment j
D	last segment played before next download
E_{tail}	energy consumption per second during tail
E_{rx}	energy consumption of bit rate
$\mathbb{E}[B_{waste}(P_2, D, n)]$	expected value (mean) of wasted data that is stored in the buffer and has not yet been viewed
$\mathbb{E}[\mathbb{E}_{\precapprox\supset\sim\approx}(\mathbb{P}_{\gg}, \mathbb{P}_{\not\ll}, \mathbb{D}, \ltimes)]$	expected value (mean) of wasted energy that was used to download content that has not yet been viewed

$$
\begin{aligned}
&bw^{+}(P_1, P_2, n, tl) = && (4.3)\\
&r_{dl}(P_0) \cdot (1-er)^{n-1} \cdot 0.3 && \text{if } 0 \leq bl^{+} < 0.15\\
&r_{dl}(P_0) \cdot (1-er)^{n-1} \cdot 0.5 && \text{if } 0.15 \leq bl^{+} < 0.35\\
&r_{dl}(P_0) \cdot (1-er)^{n-1} && \text{if } 0.35 \leq bl^{+} < 0.5\\
&r_{dl}(P_0) \cdot (1-er)^{n-1} \cdot (1 + 0.5 \cdot bl_i) && \text{if } 0.5 \leq bl^{+} < 1
\end{aligned}
$$

$$
\begin{aligned}
&bw^{-}(P_1, P_2, n, tl) = && (4.4)\\
&r_{dl}(P_0) \cdot (1-er)^{n-1} \cdot 0.3 && \text{if } 0 \leq bl^{-} < 0.15\\
&r_{dl}(P_0) \cdot (1-er)^{n-1} \cdot 0.5 && \text{if } 0.15 \leq bl^{-} < 0.35\\
&r_{dl}(P_0) \cdot (1-er)^{n-1} && \text{if } 0.35 \leq bl^{-} < 0.5\\
&r_{dl}(P_0) \cdot (1-er)^{n-1} \cdot (1 + 0.5 \cdot bl_i) && \text{if } 0.5 \leq bl^{-} < 1
\end{aligned}
$$

As we consider optimistic and pessimistic bandwidth, we also need an optimistic and pessimistic buffer level estimation. Equation 4.5 calculates the expected pessimistic buffer level, while Equation 4.6 calculates the expected optimistic buffer level. Both consist of three parts: the first part sums the already buffered time. The second part sums additionally buffered time after the download is completed, in respect to the pessimistic/optimistic bandwidth estimation. The third part represents the tail time, which is spent before the download started. For the video encoding rate $br(Q(i))$ of quality $Q(i)$ of segment i, the buffer level is determined as

$$bl^{+} = \sum_{i=P_1}^{P_2} t_i + \sum_{i=P_2+1}^{P_2+n} t_i \cdot (1 - \frac{br(Q(i))}{bw^{+}}) - tl \tag{4.5}$$

$$bl^{-} = \sum_{i=P_1}^{P_2} t_i + \sum_{i=P_2+1}^{P_2+n} t_i \cdot (1 - \frac{br(Q(i))}{bw^{-}}) - tl. \tag{4.6}$$

The equations determine an optimistic and pessimistic bandwidth and buffer estimation. Equation 4.7 contains multiple constraints for optimistic and pessimistic bandwidth and buffer estimation. The buffer constraint $bl^{+} > 0 \wedge bl^{-} < 1$ checks, whether the buffer may run empty or full. The first bandwidth constraint $bw^{+} \geq br(Q)$ checks, whether the estimated bandwidth is not less than the quality level's bit rate. The second bandwidth constraint $bw^{-} < br(Q+1)$ checks, whether the optimistic bandwidth estimation is not bigger than the next quality level's bit rate. Both constraints must be met to ensure a smooth playback.

The abandonment probability $p_{abd}(j)$ is the probability that a user abandons the video right before starting to watch segment j. The algorithm starts by selecting the quality level before the algorithm enters three interleaved loops. The first loop iterates over the segments, which can be downloaded. The loop begins at the first not yet buffered segment and ends after the limiter is exceeded. The second loop also begins at the first, not yet buffered segment and ends at the current position of the first loop. The third loop iterates over the buffered segments. Then the algorithm calculates the expected buffer state for each combination. The expected buffer state is determined using the expected buffer state of the download, which downloads all previous segments, or the current buffer state, if there is no previous download. When the user abandons a video, the data that is stored in the buffer but has not yet been viewed, is wasted. The expected (mean) wasted data $\mathbb{E}[E_{waste}]$ depends on the abandonment probability $p_{abd}(j)$ and the current buffer state $t_i \cdot br$, compare Equation 4.8. The expected wasted energy depends on the energy spent downloading data that will not be viewed on tail

energy, compare Equation 4.9. If there are no constraint violations the buffer state with the lowest expected wasted energy is stored. Per segment one buffer state is stored. The comparison is performed to the stored buffer state, which download ends at the same segment. After all combinations are evaluated, the algorithm performs a back trace. Therefore, the algorithm starts at the stored expected buffer state, which downloads the last segment. The next expected buffer is searched, which ends before the buffer state's first downloaded segment. This process continues, until the back trace reaches the current buffer. The traced downloads are returned as schedule in reversed order together with the determined quality level. The download schedule can be incomplete, which should not affect the overall behavior, because the schedule must be reevaluated after each download. If there is no optimal download without adaptation or stalling, the algorithm falls back by downloading the next segment at the selected quality level. Note, with Equations 4.8 and 4.9 we extend the original model for normal video streaming, published in [202], to adaptive streaming.

$$con = \begin{cases} 1, & \text{if } bw^+ \geq br(Q) \wedge bw^- < br(Q+1) \\ & \quad \wedge bl^+ > 0 \wedge bl^- < 1 \\ 0, & \text{else} \end{cases} \tag{4.7}$$

$$\mathbb{E}[B_{waste}(P_2, D, n)] = \sum_{i=D+1}^{P_2+n} \sum_{j=D+1}^{i} p_{abd}(j) \cdot t_i \cdot br \tag{4.8}$$

$$\mathbb{E}[E_{waste}(P_m, P_2, D, n)] = \tag{4.9}$$
$$E_{tail} \sum_{i=P_1}^{D} \max(t_i, t) + \frac{\mathbb{E}[B_{waste}(P_2, D, n)]}{r_{dl}} \cdot E_{rx}(r_{dl})$$

As shown in Figure 4.11, the adaptation strategy does not keep the same buffer size and the RRC transits to the lower state for several times. The buffer fluctuates between 10 and 30 seconds. According to the audience retention, the buffer is

Algorithm 1: Energy-Efficient Batch Adaptation (EE)

Data: $P_1, P_2, buffersize$
// $buffersize$ is the maximum buffer size
Result: $S, quali$
$E_n(P_2) = 0$
$P_1(P_2) = P_1$ // first segment in buffer
$P_2(P_2) = P_2$ // last segment in buffer
forall $q = P_1$ *to* P_2 **do**
  if $br(p) < minbw$ **then**
    $quali = q(p)$ // quality of last segment which has lower bit rate than bandwidth
  end
end
forall $i = P_2 + 1$ *to* $P_2 + buffersize$ // segments that may be downloaded without overextending the buffer size
do
  forall $j = P_2 + 1 : i$ // subsets of consecutive segments beginning with the next segment
  do
    forall $k = P_1 - 1 : j$ // segments of buffer and subset
    do
      $E_{cur} = E_n(j) + \mathbb{E}[E_{waste}(P_1(i), P_2(i), k, i - j)]$ // expected wasted energy when downloading segments until j, considering abandonment rate and tail energy
      $P_{minn} = P_1(j - 1)$
      forall $l = P_1(j - 1) : P_2(j - 1)$ **do**
        $P_{minn} += (t_l - (t_l \cdot enc/dl))$
      end
      $P_{minn} -= tail$
      if $E_{curr} < E_n(i) \wedge con(P_{minn}, j - 1, i - j, tail, quali) == 0)$ // check if expected wasted energy will be lower compared with downloading one less segment and if buffer and bandwidth constraints are fulfilled
      then
        $E_n(i) = E_{cur}$
        $lastchange(i) = j$
        $P_2(i) = j - 1$
        $P_1(i) = P_{minn}$
        $P - x(i) = i$ // schedule segments $P_2 + 1$ to j to be downloaded as a batch
      end
    end
  end
end
$end = PL_x + buffersize$
while $(end > PL_x)$ **do**
  $S = (end - lastchange(end), S)$
  $end = lastchange(end)$
end

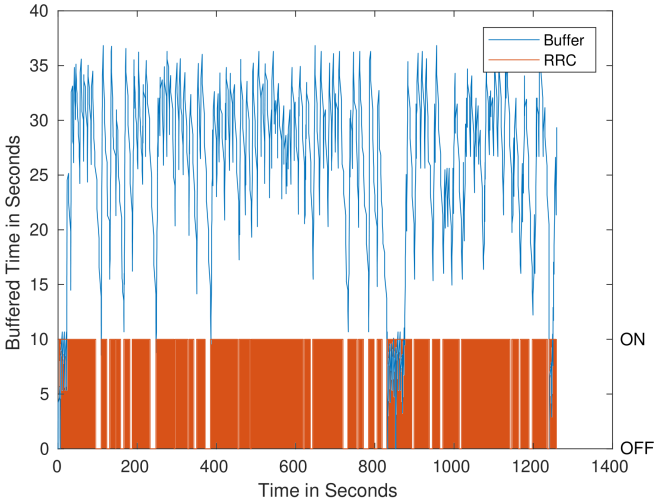


Figure 4.11: Video browsing session in which five videos are watched of which three are abandoned early. At the beginning of a video the abandonment rate is high, so the buffer is kept low to save data. The RRC state is turned off frequently to save energy with EE.

filled and depleted. During the depletion phase, the RRC transits to the lower state. After a long tail duration, the adaptation strategy continues the download to avoid stalling. Another aspect of the buffer behavior is that the buffer stays small during the first few seconds of playback. The audience retention statistics usually possess high abandon probabilities at the video's beginning. Therefore, the first segments are carefully downloaded. This approach shows the desired buffer and RRC behavior, which potentially saves energy and traffic.

4.4.2 Energy Model

Huang et al. [204] investigated the performance and power characteristics of LTE networks. They measured the energy consumption and timing of 3G, LTE and

WLAN interface in smart phones. They found that LTE possesses the highest tail time and power consumption. The tail refers to the period where the user device is still in a connected state but has finished its communication. WLAN has the smallest tail time and energy consumption. Additionally, they investigated the send and receive power consumption. They conducted several experiments and developed an energy consumption model. Also, LTE possesses the biggest base power consumption, but the least power consumption per download throughput. WLAN possesses the least amount of base energy consumption, but the biggest energy consumption per download throughput. The tail energy can be limited using Fast Dormancy (FD) [205]. As the model does not provide a specification for the 3G wireless interface with FD, we specify the missing model using the FD configuration from [202]. The FD timer is set by the device to reduce the tail time[9][10]. To reduce the tail time, we set the FD timer to 5 seconds. We use the same 3G power model to simulate the same device, which supports FD. Further, the tail time is reduced to the half of the LTE wireless interface's tail time. Li et al. [185] investigated the energy consumption of video decoding in smartphones. They conducted several experiments, watching different videos and built an empirical model of the video decoding power consumption. They compared the energy consumption with the empirical model, which shows less than 10 percent error. For our evaluation, we rely on these energy models provided for the 4G EVO and the Galaxy S. The smartphones both possess a display resolution of 800×480. Videos with a smaller or greater resolution are up-scaled or down-scaled to fit the screen size which leads to an impact on the parameters in the energy model. The different devices possess a difference in energy consumption of 300 mW for smaller and 500 mW for larger videos. Modern devices with much greater resolution than used in our evaluation should therefore be more efficient at decoding videos. For such devices, the parameters of the energy models can be simply replaced to obtain results representing current developments.

[9] https://www.gsma.com/newsroom/wp-content/uploads/2013/08/TS18v1-0.pdf
[10] http://www.3glteinfo.com/fast-dormancy-in-3gpp/

4.4.3 Audience Retention Model

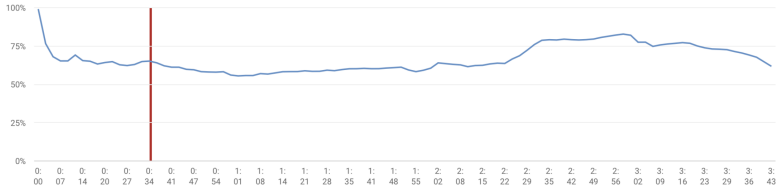


Figure 4.12: Example for viewer abandonment statistic: audience retention of video segments compared with total views of YouTube video ABSIFBFIOS. Abandonment is typically high at the beginning. An increase in the curve indicates that users skipped ahead in the video.

Most studies, such as [206], consider a video to be completely watched when benchmarking adaptation algorithms, but on actual video platforms users frequently interact with videos. Users abandon the video playback or skip some playback and resume at another playback position. Statistics about the user's video playback behavior show that users skip playback time and abandon the playback. For example, YouTube provides audience retention statistics[11] to the owner of the video channel. These statistics describe which part of a given video is watched by what share of users. Figure 4.12 shows an example for a video, which possesses a duration of 223 seconds. The audience retention starts at 100%, indicating that all users started to watch the video at its beginning. About 30% of all users abandoned the video or skipped ahead during the first 7 seconds. Due to the user's possibility to skip playback before the start of playback, the audience retention can also start with less than 100%. Users, who skip backwards to the beginning of the video, result in higher audience retention than 100%. Forward and backward skipping lets the audience retention rise and fall during the playback time.

Video segments with a low user retention, are watched by few users. Always downloading them results in a lot of wasted data and energy. We make use of

[11] https://support.google.com/youtube/answer/1715160?hl=en

this fact by delaying the download of these segments until necessary. This leads to a lower expected data consumption and energy consumption during video streaming. The details of this approach are described in Algorithm 1.

4.4.4 Results

Methodology

The simulation is based on a queue in which four different events can occur: the arrival of a video segment, the playback of a video segment, the abandonment of the video by the user, or skipping forward in the video by the user. In a single run of a simulation a single user is processed who watches five videos in a row, compare Figure 4.11. The arrival of video segments occurs according to deterministic processes based on the video bit rate and the goodput traces. For the video bit rate, we downloaded 21 popular videos every available resolution and decode them into their frame sequence using *ffprobe* to determine the location of key frames which we use as segment start in our simulation. For the goodput, we use three kinds of goodput traces: constant bandwidth, real vehicular 3G traces [175], real vehicular LTE traces [169]. The 3G traces were recorded while moving through Norway with the following type of transportation: bus, metro, tram, ferry, car, and train. The LTE traces were recorded while moving through Ghent, Belgium with the following type of transportation: bicycle, bus, car, foot, train, tram. The probability that the user abandons the video or skips ahead in the video is given through the viewer abandonment statistic for the video he is currently watching and will be described below in detail. However, such statistics are not publicly available, but only to the owner of the YouTube channel on which the corresponding video is featured. To obtain these statistics, we asked several YouTube channels to make screenshots of the audience retention statistic of their most popular videos and send them to us. Six channels replied and send us data on a total of 21 videos. We do not consider skipping back to a previous position of the video since it is not possible to determine from viewer statistics. If a video is finished or abandoned, the next video is selected randomly from the

remaining pool of videos until five videos have been viewed by the user. Each experiment was repeated 100 times. All confidence intervals in the figures are given with 95% confidence.

To generate an abandon/skip event, we first split the audience retention curve (c.f. Figure 4.12) into segment sized sections and search for the next decreasing section of the curve. We then generate a uniformly distributed random number r between 0 and 1. If r is lower than the relative decrease in viewers during this section, we generate and abandon/skip event. After such an event occurs, the remainder of the audience retention curve is searched for increasing sections. Let us define the value of the retention at the beginning of such a section as x. If such a section is found, a new uniformly distributed random number q between 0 and $1 - x$ is generated. If q is smaller than the relative increase in viewers during this section, the abandon/skip event becomes a skip event. Then the process repeats and we search for the next abandon/skip event during the same video. If no increasing section is found, the abandon/skip event becomes an abandon event and the next video is initiated. While this method does not replicate the real user behavior perfectly well, it converges towards the original audience retention curve.

Impact of the Buffer Size

First, we investigate the impact of the buffer size on QoS, energy consumption and data consumption for our new algorithm. The minimum buffer size determines the lower threshold that should never be underpassed to avoid stalling. A minimum buffer size lower than 5 seconds leads to more frequent stalling events, as can be seen in Figure 4.13. A larger minimum buffer size also leads to higher average quality and fewer quality switches. However, a higher min buffer means that more data will be wasted in the case of an abandonment event and more energy is used since idle periods cannot last as long, compare Figure 4.14. We therefore use a balanced min buffer size of 15 seconds to avoid too many stalling events.

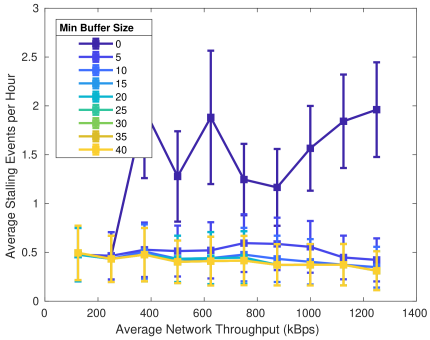


Figure 4.13: Impact of the minimum buffer size on the frequency stalling events for constant bandwidth.

The (maximum) buffer size defines how much video content may be downloaded into the buffer at most before it is played. In Figure 4.15,4.16 and 4.17 we see the impact of the buffer size on energy consumption and wasted data. A large buffer of 80 s means that we can download a lot of video content into the buffer and then pause the download until the buffer is almost depleted. This pause is very energy efficient since the idle RRC state can be reached for a long period of time. However, if the user abandons the video when the buffer is filled up, a lot of data is wasted. This means that the buffer size is a trade-of between energy and data. Since it depends on the scenario whether data or energy is more valuable, we use three parameters for the buffer size in the following: 30s, 50s, 80s.

Impact of the Network Type on Saved Energy

Next, we investigate how much energy can be saved with EE compared to KLU in different scenarios. In Figures 4.15, 4.16, and 4.17 the x-axis shows the average throughput of the different network traces that we use, while the y-axis shows the average amount of energy that is consumed per second by the video download or the average wasted traffic per scenario. In a WiFi scenario (Figure 4.15) the

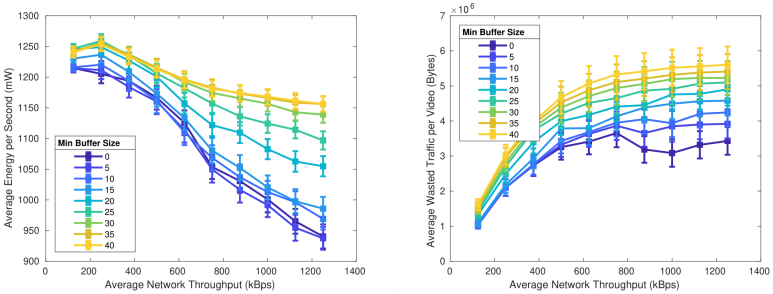


Figure 4.14: Impact of the minimum buffer sized on energy and data saved for constant bandwidth using an LTE interface.

consumed energy can be reduced by about 25% with a large buffer. While less data is wasted with EE than with KLU, data is usually not limited in WiFi settings compared with other mobile settings where data is limited through data plans. In conclusion, EE leads to higher energy efficiency in a WiFi setting. In a 3G scenario (Figure 4.16) EE leads to $10 - 12\%$ of energy saved when comparing a 30s buffer to an 80s buffer. In terms of data, it becomes visible that especially for high buffer sizes, EE is much more efficient than KLU since it makes use of user abandonment statistics. In absolute numbers however, in average only about 2MB are saved for each video that is started. In contrast, savings are much higher in LTE networks (Figure 4.17). For an 80s buffer, we can reduce the average energy consumption by $35 - 40\%$ and the wasted traffic by about 50%. Here, the saved data is larger, but also only lies between 3-6 MB which corresponds to 12-24 seconds of 720p video content. From the energy perspective it makes the most sense to use EE in LTE scenarios where it can result in much longer battery time. Furthermore, it is visible that a large buffer always increases the battery time significantly while the saved data is insignificant. Even with a large buffer only little data is wasted since the buffer is not filled up during scenes in the video where many users abandon.

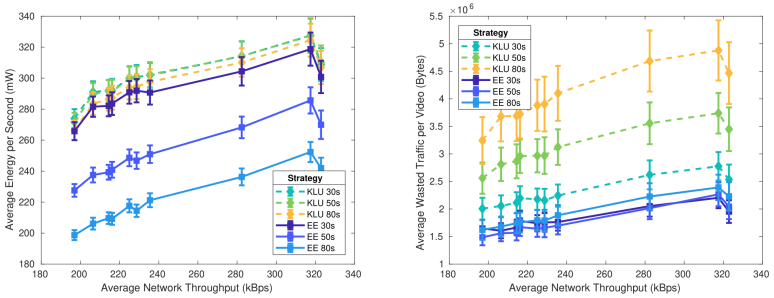


Figure 4.15: Resourcefulness of KLU and EE in constant WiFi scenario

Performance in Terms of Application-Layer QoS

Next, we compare the QoS for KLU and EE in the LTE scenario. The average playback quality is only about 1% higher in KLU, c.f. Figure 4.18. With KLU the average number of quality switches per hour lies between 7 and 50 depending on the scenario and is about 50% - 100% higher than with EE. The number of stalling events is higher for EE with 0.4 stalls per hour because KLU constantly stays at high buffer, while EE has ON/OFF phases to conserve energy. An exception can be observed for the subway scenario. In this scenario, the user passes through many long tunnels where no connection to the network is possible. If such a tunnel cannot be predicted, stalling can only be avoided if the buffer is very high when we enter the tunnel. So, only KLU with 80 s buffer size can avoid some stalling events. Notice, that we already discussed the optimization of HAS in such scenarios in 3.4.1. In such a situation, only a very large buffer can reduce the number of stalling events. However, since a large buffer leads to more wasted traffic, the best-case scenario would be to predict such a rare outage event and to temporarily increase the buffer in time.

To sum up the results, we see that EE is much more resourceful than KLU while maintaining a similar QoS on application layer. An exception is only observed in a scenario with very bad connectivity where energy and data efficiency are a

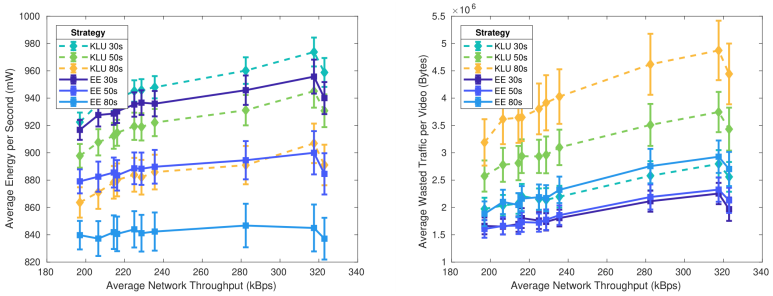


Figure 4.16: Resourcefulness of KLU and EE in vehicular 3G scenarios

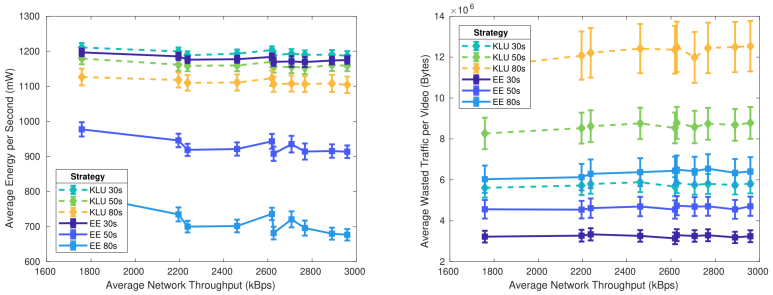
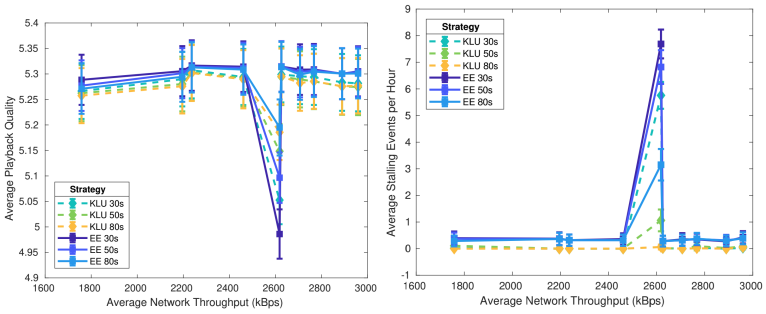


Figure 4.17: Resourcefulness of KLU and EE in vehicular LTE scenarios

secondary concern. When such a scenario is detected, the player should switch to a conservative adaptation strategy that maintains a high buffer such as KLU. To detect such a scenario, it is imaginable to collect data on the QoS combined with the devices location. A train ride, for example, could then be easily be mapped to a bandwidth curve which would make bandwidth estimations much easier.



(a) (b)

Figure 4.18: Application layer QoS for KLU and EE in a vehicular LTE scenario

4.5 Lessons Learned

The popularity of video streaming has increased considerably on mobile devices in the past years. Consumers do not only care about the quality of the content and the quality of the service, but also about the energy efficiency and the data efficiency of their device while they use the service. In this chapter, we discussed how data and energy can be used more efficiently in mobile video streaming where these are limited resources. We identified adaptation algorithms as a good point in the service chain for optimizing the expected energy consumption and the expected data consumption.

To answer the question to what degree WiFi-caches can be used to save traffic by avoiding redundant downloads within the same local network, we have conducted two sets of experiments. The first set is comprised of load tests in a testbed where we investigate the impact of a cache hit or miss on the loading time of popular websites. In addition, we analyze the performance impact of the number of clients that are simultaneously connected to the cache. As a second experiment, we set up caches at two public WiFi hotspots in different locations and were able to derive interesting observations regarding the ratio

of encrypted — and therefore uncacheable — traffic that seem to confirm the recent developments described above. In addition, we propose a queueing model for caches that we use to conduct a mean value analysis and compare it to the results from the measurement study. We observed that a large amount of the traffic was encrypted with HTTPS and could not be cached without using man in the middle attacks. As a consequence, using a router-cache did not lead to a significant amount of saved traffic.

To optimize adaptive streaming algorithms, it is necessary to know more about the user activity during video streaming. However, this is a field that is researched little. In this chapter, we discussed the user behavior during video streaming which was measured over multiple years using the YoMoApp. The measurement results show that many user actions are correlated. For example, stalling, pausing and seeking often occurs in similar ratios. We found that mobile users are less likely to abandon a video due to stalling than non-mobile users. Particularly interesting is the fact, that *leaving full screen* is an indicator that users will soon abandon the video, which could be used to save data.

Furthermore, we combine an adaptive streaming heuristic and an energy efficiency scheme for video streaming using user behavior statistics into a new adaptive streaming heuristic. Furthermore, we extend the heuristic with a data efficiency scheme by comparing the abandonment probability with tail energy that can be saved to determine a schedule for downloading the next segments. We investigate the performance of our heuristic by simulating a video session where users watch five consecutive videos. Simulated users abandon videos or skip through parts of a video based on viewer retention statistics of real YouTube videos and users. Our results show that being aware of the user behavior and scheduling segment downloads efficiently can reduce the energy consumption of the video download by about 25% in WiFi networks and by $35 - 40\%$ in LTE networks with a buffer size of 80s. The wasted traffic that results from video segments that are downloaded but never viewed can be reduced by about 50% in the latter scenario. The gain in terms of energy and data comes at no cost in

terms of quality and stalling compared with the baseline KLUDCP, except for scenarios with very long connection interruptions.

5 Conclusion

As one of the most popular Internet applications, video streaming will account for 79% of all Internet traffic in 2020 [28]. The type of video streaming that is currently mostly used is HTTP Adaptive Streaming. ISPs and video service providers try to mitigate stalling and increase the delivered video bit rate to meet user expectations and maximize the QoE. In particular in mobile networks, where the bandwidth is fluctuating, it is difficult to keep the video quality high without stalling. It is therefore important to know the interplay between video player parameters, network bandwidth, content and transmission method. A further aspect is the high data and energy consumption of mobile devices when playing videos. As ISPs mostly give customers mobile access with data caps and the battery capacity of modern devices increases slowly, these are valuable resources that are consumed rapidly when watching Internet videos. Conserving the resources of the user device is therefore another goal in mobile scenarios.

In this monograph, we study the performance of video streaming in mobile networks. While our results also apply to non-mobile video streaming, the mobile case is more interesting since it is much harder to fulfill user expectations. We show that stalling, quality switches, and resource consumption can be reduced while the video quality can be increased significantly by using different adaptation techniques. For this, we use simulations, measurement studies, and analytical models. The major insights of this monograph are that there are many parameters in video streaming that can easily be fine-tuned to reach the optimal configuration for a given scenario. The more complex the video streaming application becomes, from classical streaming to adaptive streaming to 360° video streaming, the more parameters are added. This also means, that the more complex such an

application is, the greater the difference between standard solutions and optimal solutions becomes. Another important aspect of this work is how we link the QoE to user specific behavior and engagement. Not only do we determine a strong correlation between the QoE and the User Engagement, but we identify the scenario in which a video is viewed as a significant indicator as to which QoE model should be applied. In addition, we make use of user data such as head movements and audience retention to increase the video quality and to reduce the data consumption in video streaming. Finally, we find that QoE fairness, which is currently ignored by most video service providers and ISPs, can be realized without decreasing the average QoE.

As a first step, we investigate how different buffer settings and policies impact stalling duration and frequency using analytical models and empirical measurement studies. This also gives us an insight into the user's QoE and engagement. We are able to show that the buffer dimensions must be selected according to the user scenario. For example, for video browsing we recommend to start playing a video after 1-2 s of video content is downloaded while this value lies at 2-4 s for normal video streaming. In scenarios with high bandwidth, we recommend to limit the buffer size to avoid wasted traffic in the case of abandonment. Furthermore, we find a strong correlation between the QoE and the User Engagement in video streaming.

In the third chapter, we consider adaptation related aspects besides the buffer that can be optimized in HAS. We start with an analytical approach which allows us to derive KPIs of HAS. The remainder of the chapter presents quadratic programs that model HAS in detail and optimize video quality, switches and stalling. In an investigation of YouTube download traces, we observe that 95% of stalling events could have been avoided with different adaptation decisions while keeping the initial delay below 10 s. We demonstrate that it is possible to reduce the number of switches of an optimal solution strongly while only sacrificing little in term of video quality. Using three approaches to fairness, we show that the video quality can be increased by 0.29 quality layers in average while resulting in fairer QoS on the application layer. We adapt our optimization

approach to 360° and use it to identify the potential traffic savings that could be achieved using three different approaches to viewport prediction. Our results show that a statistics-based viewport prediction should be used in VODs and a extrapolation-based approach should be used for live streaming. Machine learning-based approaches for viewport prediction still require more research.

The fourth chapter focuses on energy and data conservation in mobile networks. As a first approach we investigate the degree to which Internet traffic be cached at WiFi hotspots. Identifying that most videos are encrypted during transmission and that there is little anyone besides the video service provider can do to cache these videos at the edge, we turn our gaze towards other solutions to reduce the data consumption in video streaming. As such, we evaluate the behavior and interactions of mobile YoMoApp users and detect that many user interactions are correlated. For example, users who leave the full screen mode of a video are very likely to abandon the video within seconds. This can be used to limit the amount of new video segments that are downloaded into the buffer, after leaving full screen, thus reducing the average wasted traffic. We then develop a new adaptation algorithm that uses video abandonment statistics to conserve data and energy in mobile networks. in a simulation, we were able to show that the algorithm is able to save up to 25% of energy in WiFi networks, and $35 - 40\%$ of energy and 50% of wasted traffic in LTE networks compared to another adaptation algorithm. At the same time, it does not perform significantly worse in terms of video quality and stalling, except for a single scenario.

For practical purposes, our results show that with changes at the video client QoE can be increased while being more resource efficient. An increase in QoE and a reduction in energy consumption means that users are likely to spend more time on video platforms which is in the best interest of video service providers. Since most users use the standard client offered by the video service provider, it is the providers responsibility to implement changes we recommend in this work.

For further research, this work is an important milestone for the optimization of QoE, not only in video streaming but also in applications that use video streaming

as their foundation, such as video conferencing, virtual reality, augmented reality, and cloud gaming. Future work could extend our optimization problems to include such technologies and adapt our analytical models for them. In the future new technologies will emerge that will be even more data intensive than video streaming. For these applications similar approaches will be necessary as we have presented in this work. For applications with low latency requirements, it is very important to develop optimized adaptation algorithms in order to deliver video streaming with high user perceived QoE.

A Methodology of Section 3.5.1

In this measurement study, we developed a testbed in which multiple clients use a shared bottleneck. The implementation is done using virtual machines in order to have a flexible setup. So, the number of clients is only restricted by the available hardware, especially memory and processing capacity. We were able to run up to four clients in parallel. The clients are running Ubuntu Linux with a Chromium web browser. With little effort, the clients could be replaced with arbitrary devices running a standard web browser, for example smart phones. The test setup consists of multiple clients that all share the same bottleneck. A control-server manages the experiments, provides the configurations of the clients and collects the results. The bottleneck is implemented via a bandwidth limiter that connects the clients and the control server to the Internet with a bandwidth of $1\,\mathrm{Gbit/s}$.

As in the YouTube Player the video source cannot be changed, we use a video directly from YouTube[1]. YouTube provides videos in different video codecs and container formats (e.g., 3GP, VP9, WebM) and quality levels, in order to give suitable versions for different devices [207]. In our case, the browser Chromium selected only the VP9 version of the test video. Furthermore, videos are transmitted via QUIC over UDP instead of TCP in Chromium. As we are performing a steady-state analysis, we use a long video. It has duration of 64 minutes, and comes in six different quality representations, going from the lowest quality 144p to full HD resolution 1080p. For more information on the quality levels see Table A.1.

[1] https://www.youtube.com/watch?v=7ojfoBmR1fw

Table A.1: Quality levels of the test video. All videos come with 30 frames per second. In our test video, the video and audio are stored independently. The Itag identifies the resolution and bit rate of the YouTube video.

Itag	Resolution	Bit rate	File size
249	audio (50k)	53 kbit/s	20.81 MB
250	audio (70k)	72 kbit/s	27.54 MB
251	audio (160k)	139 kbit/s	55.13 MB
278	144p	113 kbit/s	38.11 MB
242	240p	224 kbit/s	72.35 MB
243	360p	413 kbit/s	140.39 MB
244	480p	764 kbit/s	207.96 MB
247	720p	1515 kbit/s	279.89 MB
248	1080p	2656 kbit/s	390.96 MB

Table A.2: Characteristics of the bandwidth scenarios given in kB/s.

Scenario	Mean	Std. Dev.	$10^{th}/90^{th}$ Perc.	Min/Max
Bus	244.7	164.1	34.6/499.6	25.3/773.3
Tram	100.4	62.4	23.2/169.3	53.5/824

The bandwidth is limited using bandwidth traces, which reflect real-world commuting scenarios using different means of transportation. We use two different bandwidth traces, both recorded by Riiser et al. using a notebook and a 3G modem to determine the download speed a GPS module to determine the location[92]. The first bandwidth trace was recorded during traveling with a bus. This scenario has an average bandwidth of 251 kB/s. The second bandwidth trace was recorded while commuting with a tram, it has a significantly lower average bandwidth of 103 kB/s. Details about these bandwidth scenarios can be found in Table A.2. These bandwidth traces are replayed using the built-in tool

Table A.3: Bandwidth configuration of the experiments. In case of variable bandwidth, the bandwidth scenario is multiplied with the given factor. For constant bandwidth, the average of the bandwidth scenario is used.

# of clients	Browser	Bus		Tram	
		Var.	Const.	Var.	Constant
1	Chromium	0.5	122.3 kB/s	0.5	50.2 kB/s
2	Chromium	1	244.7 kB/s	1	100.4 kB/s
3	Chromium	1.5	367.0 kB/s	1.5	150.6 kB/s
4	Chromium	2	489.4 kB/s	2	200.8 kB/s

tc from the package *iproute2*. Mobile scenarios are of special interest, because it is difficult for the video players to adapt to the varying bandwidth. For comparison, we repeat all experiments with a constant bandwidth, which is exactly the average of the bandwidth scenarios. In order to ensure comparability, the bandwidth is depending on the number of simultaneous clients. This way, each client has the same bandwidth in all experiments. This means, that the video quality with multiple clients should be at least the same as in case of a single client. The detailed configuration of all experiments is listed in Table A.3.

The procedure of the experiments is as follows: The clients simultaneously start to play the same video using the standard YouTube video player embedded in a website. The window of the YouTube player has a width of 1920px, so that the video player will choose all available video resolutions up to 1080px. In previous experiments we found, that the YouTube player does not select high video resolutions if the window size is too small. During the playback, the client records the behavior of the video player. As soon as the playback has finished, this information is transmitted to the Control-Server. Then, the playback of the video is restarted. The cache of the web browser is deactivated, so that the video has to be completely retransmitted. During the playback, the Bandwidth-Limiter

controls the throughput according to the current bandwidth scenario. When it has reached the end of the bandwidth trace, it starts all over again.

Table A.4: Number of repetitions for each experiment. As the players play independently, in one experiment there are different repetitions for each player.

Scenario	# of players	# of runs	
		Var. bandw.	Const. bandw.
Bus	1	131	11
Bus	2	95-96	46
Bus	3	18-20	23
Bus	4	12	19
Tram	1	28	19
Tram	2	27	45
Tram	3	57	18
Tram	4	19	39

In order to get statistics about the behavior of the video players, the playback is repeated numerous times. The exact experiments and their number of repetitions is listed in Table A.4. Due to stalling, some players can have more repetitions than others. This span is also indicated in this table. Using the JavaScript API of the YouTube player, we collected the following information for each player:

- the number and duration of stalling events,
- the quality level in which the video was played and
- the number of quality adaptation events

B Coefficients of Correlation for Section 4.3

coef. of corr.	user eng.	stall ratio	stalls	stall dur.	full screen	rotations	seeking
user eng.	-	-0.0828	-0.0199	-0.0113	0.0967	0.1502	-0.1018
stalling ratio	-0.0828	-	**0.9520**	**0.9782**	-0.0109	-0.0099	0.0821
stalls	-0.0199	**0.9520**	-	**0.9484**	0.0240	0.0292	0.1303
mean stall dur.	-0.0113	**0.9782**	**0.9484**	-	0.0092	0.0173	0.0742
full screen	0.0967	-0.0109	0.0240	0.0092	-	**0.6957**	0.0670
screen rotations	0.1502	-0.0099	0.0292	0.0173	**0.6957**	-	0.0261
seeking	-0.1018	0.0821	0.1303	0.0742	0.0670	0.0261	-

Table B.1: Spearman coefficient of correlation between interesting measures. Values greater than 0.2 *are marked boldly.*

Bibliography and References

Bibliography of the Author

Book Chapters

[1] E. Liotou, T. Hoßfeld, C. Moldovan, F. Metzger, D. Tsolkas, and N. Passas, "The value of context-awareness in bandwidth-challenging HTTP adaptive streaming scenarios," in *Autonomous Control for a Reliable Internet of Services*, Springer, Cham, 2018, pp. 128–150.

Journal Papers

[2] 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, pp. 211–224, 2016.

[3] F. Metzger, E. Liotou, C. Moldovan, and T. Hoßfeld, "TCP video streaming and mobile networks: Not a love story, but better with context," *Computer Networks*, vol. 109, pp. 246–256, 2016.

Conference Papers

[4] T. Hoßfeld, C. Moldovan, and C. Schwartz, "To each according to his needs: Dimensioning video buffer for specific user profiles and behavior," in *2015 IFIP/IEEE International Symposium on Integrated Network Management (IM)*, IEEE, 2015, pp. 1249–1254.

[5] M Seufert, V Burger, F Wamser, P Tran-Gia, C Moldovan, and T Hoßfeld, "Utilizing home router caches to augment CDNs toward information-centric networking," in *European Conference on Networks and Communications (EuCNC), Paris, France*, 2015.

[6] C. Moldovan and F. Metzger, "Bridging the gap between QoE and user engagement in HTTP video streaming," in *2016 28th International Teletraffic Congress (ITC 28)*, IEEE, vol. 1, 2016, pp. 103–111.

[7] C. Moldovan and T. Hoßfeld, "Impact of variances on the QoE in video streaming," in *Teletraffic Congress (ITC 28), 2016 28th International*, IEEE, vol. 3, 2016, pp. 19–24.

[8] C. Moldovan, C. Sieber, P. Heegaard, W. Kellerer, and T. Hoßfeld, "YouTube can do better: Getting the most out of video adaptation," in *2016 28th International Teletraffic Congress (ITC 28)*, IEEE, vol. 3, 2016, pp. 7–12.

[9] 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 *2016 IEEE International Conference on Communications (ICC)*, IEEE, 2016, pp. 1–6.

[10] C. Moldovan, K. Hagn, C. Sieber, W. Kellerer, and T. Hoßfeld, "Keep calm and don't switch: About the relationship between switches and quality in HAS," in *2017 29th International Teletraffic Congress (ITC 29)*, IEEE, vol. 3, 2017, pp. 1–6.

[11] C. Moldovan, F. Metzger, S. Surminski, T. Hoßfeld, and V. Burger, "Viability of Wi-Fi caches in an era of HTTPS prevalence," in *2017 IEEE International Conference on Communications Workshops (ICC Workshops)*, IEEE, 2017, pp. 1370–1375.

[12] M. Seufert, C. Moldovan, V. Burger, and T. Hofeld, "Applicability and limitations of a simple WiFi hotspot model for cities," in *2017 13th Inter-*

national Conference on Network and Service Management (CNSM), IEEE, 2017, pp. 1–7.

[13] S. Surminski, C. Moldovan, and T. Hoßfeld, "Saving bandwidth by limiting the buffer size in HTTP adaptive streaming," in *MMBnet 2017*, 2017, pp. 5–21.

[14] C. Moldovan, L. Skorin-Kapov, P. E. Heegaard, and T. Hoßfeld, "Optimal fairness and quality in video streaming with multiple users," in *30th International Teletraffic Congress (ITC 30)*, IEEE, vol. 1, 2018, pp. 73–78.

[15] S. Surminski, C. Moldovan, and T. Hoßfeld, "Practical QoE evaluation of adaptive video streaming," in *International Conference on Measurement, Modelling and Evaluation of Computing Systems*, Springer, 2018, pp. 283–292.

[16] C. Moldovan, F. Wamser, and T. Hoßfeld, "User behavior and engagement of a mobile video streaming user from crowdsourced measurements," in *2019 Eleventh International Conference on Quality of Multimedia Experience (QoMEX)*, IEEE, 2019, pp. 1–3.

[17] A. Schwind, L. Janiak, C. Moldovan, F. Wamser, and T. Hoßfeld, "Peeking under the hood: How the measurement setup influences the video streaming behavior," in *2019 22nd Conference on Innovation in Clouds, Internet and Networks and Workshops (ICIN)*, IEEE, 2019, pp. 13–18.

[18] 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*, ACM, 2019, pp. 267–272.

[19] C. Moldovan, F. Wamser, and T. Hoßfeld, "Energy-Efficient adaptation logic for HTTP streaming in mobile networks," in *2019 International Conference on Networked Systems (NetSys) (NetSys'19)*, Garching b. München, Germany, Mar. 2019.

[20] C. Moldovan, F. Loh, M. Seufert, and T. Hoßfeld, "Optimizing HAS for 360-degree videos," in *AnNet*, IEEE/IFIP, 2020.

[21] F. Loh, F. Wamser, C. Moldovan, B. Zeidler, D. Tsilimantos, S. Valentin, and T. Hoßfeld, "Is the uplink enough? estimating video stalls from encrypted network traffic," in *Network Operations and Management Symposium (NOMS)*, IEEE / IFIP, 2020.

[22] A. Schwind, C. Moldovan, T. Janiak, N. D. Dworschak, and T. Hoßfeld, "Don't stop the music: Crowdsourced QoE assessment with stalling," in *2020 12th International Conference on Quality of Multimedia Experience (QoMEX)*, IEEE, 2020.

Technical Reports

[23] C. Sieber, K. Hagn, C. Moldovan, T. Hoßfeld, and W. Kellerer, "Towards machine learning-based optimal HAS," 2018.

General References

[24] J. Hosek, M. Ries, P. Vajsar, L. Nagy, S. Andreev, O. Galinina, Y. Koucheryavy, Z. Sulc, P. Hais, and R. Penizek, "User's happiness in numbers: Understanding mobile YouTube quality expectations," in *2015 38th International Conference on Telecommunications and Signal Processing (TSP)*, IEEE, 2015, pp. 607–611.

[25] F. Dobrian, V. Sekar, A. Awan, I. Stoica, D. Joseph, A. Ganjam, J. Zhan, and H. Zhang, "Understanding the impact of video quality on user engagement," *ACM SIGCOMM Computer Communication Review*, vol. 41, no. 4, pp. 362–373, 2011.

[26] R. Trestian, I.-S. Comsa, and M. F. Tuysuz, "Seamless multimedia delivery within a heterogeneous wireless networks environment: Are we there yet?" *IEEE Communications Surveys & Tutorials*, vol. 20, no. 2, pp. 945–977, 2018.

[27] Sandvine, *The global internet phenomena report*, 2018.

[28] G. M. D. T. Forecast, "Cisco visual networking index: Global mobile data traffic forecast update, 2017–2022," *Update*, vol. 2017, p. 2022, 2019.

[29] Ericsson, "Mobile traffic by application," *June*, 2020.

[30] A. Ericsson, "Ericsson mobility report," *Nov*, 2018.

[31] E Mohyeldin, "Minimum technical performance requirements for IMT-2020 radio interface(s)," in *Proc. ITU-R Workshop IMT-2020 Terrestrial Radio Interfaces*, 2016.

[32] S. M. Patrick Le Callet and e. Andrew Perkis, "Qualinet white paper on definitions of quality of experience (2012)," *European Network on Quality of Experience in Multimedia Systems and Services (COST Action IC 1003)*, 2013.

[33] 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*, IEEE, 2012, pp. 1–6.

[34] C. Alberti, D. Renzi, C. Timmerer, C. Mueller, S. Lederer, S. Battista, and M. Mattavelli, "Automated QoE evaluation of dynamic adaptive streaming over http," in *Fifth International Workshop on Quality of Multimedia Experience (QoMEX)*, IEEE, 2013.

[35] M. Z. Chowdhury, M. N. Islam, Y. M. Seo, Y. K. Lee, S. B. Kang, S. W. Choi, and Y. M. Jang, "Characterizing QoS parameters and application of soft-QoS scheme for 3G wireless networks," in *2008 10th International Conference on Advanced Communication Technology*, IEEE, vol. 1, 2008, pp. 760–764.

[36] I. Rec, "E. 800 (2008)," *Definitions of terms related to quality of service*, 2008.

[37] H. O'Brien, "Theoretical perspectives on user engagement," in *Why Engagement Matters*, Springer, 2016, pp. 1–26.

[38] M. Lalmas, H. O'Brien, and E. Yom-Tov, "Measuring user engagement," *Synthesis Lectures on Information Concepts, Retrieval, and Services*, vol. 6, no. 4, pp. 1–132, 2014.

[39] M. I. Hwang and R. G. Thorn, "The effect of user engagement on system success: A meta-analytical integration of research findings," *Information & Management*, vol. 35, no. 4, pp. 229–236, 1999.

[40] M. Zink, J. Schmitt, and R. Steinmetz, "Layer-encoded video in scalable adaptive streaming," *IEEE Transactions on Multimedia*, vol. 7, no. 1, 2005.

[41] L. Yitong, S. Yun, M. Yinian, L. Jing, L. Qi, and Y. Dacheng, "A study on quality of experience for adaptive streaming service," in *International Conference on Communications Workshops (ICC)*, IEEE, 2013.

[42] N. Cranley, P. Perry, and L. Murphy, "User perception of adapting video quality," *International Journal of Human-Computer Studies*, vol. 64, no. 8, pp. 637–647, 2006.

[43] T. Hoßfeld, L. Skorin-Kapov, P. E. Heegaard, and M. Varela, "A new QoE fairness index for QoE management," *Quality and User Experience*, vol. 3, no. 1, 2018.

[44] F. Qamar, M. N. Hindia, T. Abbas, K. B. Dimyati, and I. S. Amiri, "Investigation of QoS performance evaluation over 5G network for indoor environment at millimeter wave bands," *International Journal of Electronics and Telecommunications*, vol. 65, no. 1, pp. 95–101, 2019.

[45] R. K. Mok, E. W. Chan, and R. K. Chang, "Measuring the quality of experience of HTTP video streaming," in *IFIP/IEEE Int. Symposium on Integrated Network Management (IM)*, 2011, pp. 485–492.

[46] R. C. Streijl, S. Winkler, and D. S. Hands, "Mean opinion score (MOS) revisited: Methods and applications, limitations and alternatives," *Multimedia Systems*, vol. 22, no. 2, pp. 213–227, 2016.

[47] T. Hoßfeld, P. E. Heegaard, and M. Varela, "QoE beyond the MOS: Added value using quantiles and distributions," in *Quality of Multimedia Experience (QoMEX), 2015 Seventh International Workshop on*, IEEE, 2015, pp. 1–6.

[48] M. Seufert, "Fundamental advantages of considering quality of experience distributions over mean opinion scores," in *2019 Eleventh International Conference on Quality of Multimedia Experience (QoMEX)*, IEEE, 2019, pp. 1–6.

[49] T. Hoßfeld, R. Schatz, and S. Egger, "SOS: The MOS is not enough!" In *2011 Third International Workshop on Quality of Multimedia Experience*, IEEE, 2011, pp. 131–136.

[50] T. Hoßfeld, R. Schatz, E. Biersack, and L. Plissonneau, "Internet video delivery in YouTube: From traffic measurements to quality of experience," in *Data Traffic Monitoring and Analysis*, Springer, 2013, pp. 264–301.

[51] Q. Huynh-Thu and M. Ghanbari, "Temporal aspect of perceived quality in mobile video broadcasting," *IEEE Transactions on Broadcasting*, vol. 54, no. 3, pp. 641–651, 2008.

[52] R. K. Mok, W. Li, and R. K. Chang, "Irate: Initial video bitrate selection system for HTTP streaming," *IEEE Journal on Selected Areas in Communications*, vol. 34, no. 6, pp. 1914–1928, 2016.

[53] K. D. Singh, Y. Hadjadj-Aoul, and G. Rubino, "Quality of experience estimation for adaptive HTTP/TCP video streaming using H. 264/AVC," in *Consumer Communications and Networking Conference (CCNC), 2012 IEEE*, IEEE, 2012, pp. 127–131.

[54] T. Mäki, M. Varela, and D. Ammar, "A layered model for quality estimation of HTTP video from QoE measurements," in *Signal-Image Technology & Internet-Based Systems (SITIS), 2015 11th International Conference on*, IEEE, 2015, pp. 591–598.

[55] T. Hoßfeld, R. Schatz, and U. R. Krieger, "QoE of YouTube video streaming for current internet transport protocols," in *Measurement, Modelling, and Evaluation of Computing Systems and Dependability and Fault Tolerance*, Springer, 2014, pp. 136–150.

[56] 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*, vol. 17, no. 1, pp. 469–492, 2015.

[57] H. T. Tran, N. P. Ngoc, A. T. Pham, and T. C. Thang, "A multi-factor QoE model for adaptive streaming over mobile networks," in *Globecom Workshops (GC Wkshps), 2016 IEEE*, IEEE, 2016, pp. 1–6.

[58] Z. Duanmu, K. Zeng, K. Ma, A. Rehman, and Z. Wang, "A quality-of-experience index for streaming video," *IEEE Journal of Selected Topics in Signal Processing*, 2016.

[59] T. Hoßfeld, L. Skorin-Kapov, P. E. Heegaard, M. Varela, and K.-T. Chen, "On additive and multiplicative QoS-QoE models for multiple QoS parameters," in *5th ISCA/DEGA Workshop on Perceptual Quality of Systems, PQS 2016*, 2016.

[60] M. Fiedler, T. Hoßfeld, and P. Tran-Gia, "A generic quantitative relationship between quality of experience and quality of service," *IEEE Network*, vol. 24, no. 2, pp. 36–41, 2010.

[61] 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 (ToN)*, vol. 22, no. 1, pp. 326–340, 2014.

[62] 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," *ACM SIGCOMM Computer Communication Review*, vol. 44, no. 4, pp. 187–198, 2015.

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

[64] C. Kreuzberger, B. Rainer, H. Hellwagner, L. Toni, and P. Frossard, "A comparative study of DASH representation sets using real user characteristics," in *Proceedings of the 26th International Workshop on Network and Operating Systems Support for Digital Audio and Video*, ACM, 2016, p. 4.

[65] C. Timmerer, M. Maiero, and B. Rainer, "Which adaptation logic? an objective and subjective performance evaluation of HTTP-based adaptive media streaming systems," *arXiv preprint arXiv:1606.00341*, 2016.

[66] 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, 2017.

[67] Y. Sani, A. Mauthe, and C. Edwards, "Adaptive bitrate selection: A survey," *IEEE Communications Surveys & Tutorials*, vol. 19, no. 4, pp. 2985–3014, 2017.

[68] M. T. Diallo, F. Fieau, and J.-B. Hennequin, "Impacts of video quality of experience on user engagement in a live event," in *Multimedia and Expo Workshops (ICMEW), 2014 IEEE Int. Conference on*, IEEE, 2014, pp. 1–7.

[69] M. Z. Shafiq, J. Erman, L. Ji, A. X. Liu, J. Pang, and J. Wang, "Understanding the impact of network dynamics on mobile video user engagement," in *The 2014 ACM international conference on Measurement and modeling of computer systems*, ACM, 2014, pp. 367–379.

[70] A. Balachandran, V. Sekar, A. Akella, S. Seshan, I. Stoica, and H. Zhang, "A quest for an internet video quality-of-experience metric," in *11th ACM Workshop on Hot Topics in Networks*, ser. HotNets-XI, Redmond, Washington: ACM, 2012, pp. 97–102, ISBN: 978-1-4503-1776-4. DOI: 10.1145/2390231.2390248. [Online]. Available: http://doi.acm.org/10.1145/2390231.2390248.

[71] ——, "Developing a predictive model of quality of experience for internet video," in *ACM SIGCOMM Computer Communication Review*, ACM, vol. 43, 2013, pp. 339–350.

[72] I. Bartolec, I. Orsolic, and L. Skorin-Kapov, "Inclusion of end user playback-related interactions in YouTube video data collection and ml-based performance model training," in *2020 Twelfth International Conference on Quality of Multimedia Experience (QoMEX)*, IEEE, 2020, pp. 1–6.

[73] T. Hoßfeld, L. Atzori, P. E. Heegaard, L. Skorin-Kapov, and M. Varela, "The interplay between QoE, user behavior and system blocking in QoE management," in *2019 22nd Conference on Innovation in Clouds, Internet and Networks and Workshops (ICIN)*, IEEE, 2019, pp. 112–117.

[74] S. S. Krishnan and R. K. Sitaraman, "Video stream quality impacts viewer behavior: Inferring causality using quasi-experimental designs," *Networking, IEEE/ACM Transactions on*, vol. 21, no. 6, pp. 2001–2014, 2013.

[75] J. Lehmann, M. Lalmas, E. Yom-Tov, and G. Dupret, "Models of user engagement," in *User Modeling, Adaptation, and Personalization*, Springer, 2012, pp. 164–175.

[76] P. Casas, R. Schatz, and T. Hoßfeld, "Monitoring YouTube QoE: Is your mobile network delivering the right experience to your customers?" In *2013 IEEE Wireless Communications and Networking Conference (WCNC 2013)*, Shanghai, China, Apr. 2013.

[77] M. Jarschel, F. Wamser, T. Hohn, T. Zinner, and P. Tran-Gia, "SDN-based application-aware networking on the example of YouTube video streaming," in *Software Defined Networks (EWSDN), 2013 Second European Workshop on*, IEEE, 2013, pp. 87–92.

[78] S. Baraković and L. Skorin-Kapov, "Survey and challenges of QoE management issues in wireless networks," *Journal of Computer Networks and Communications*, vol. 2013, 2013.

[79] T. Hoßfeld, R. Schatz, M. Varela, and C. Timmerer, "Challenges of QoE management for cloud applications," *Communications Magazine, IEEE*, vol. 50, no. 4, pp. 28–36, 2012.

[80] S Moller, S. Schmidt, and J. Beyer, "Gaming taxonomy: An overview of concepts and evaluation methods for computer gaming QoE," in *Quality of Multimedia Experience (QoMEX), 2013 Fifth International Workshop on*, IEEE, 2013, pp. 236–241.

[81] M. Jarschel, D. Schlosser, S. Scheuring, and T. Hoßfeld, "Gaming in the clouds: QoE and the users' perspective," *Mathematical and Computer Modelling*, vol. 57, no. 11, pp. 2883–2894, 2013.

[82] B. J. Villa and P. E. Heegaard, "Towards knowledge-driven QoE optimization in home gateways," in *ICNS 2011, The Seventh International Conference on Networking and Services*, 2011, pp. 252–256.

[83] L. Plissonneau, E. Biersack, and P. Juluri, "Analyzing the impact of YouTube delivery policies on user experience," in *24th International Teletraffic Congress*, International Teletraffic Congress, 2012, p. 28.

[84] K. De Moor, M. R. Quintero, D. Strohmeier, and A. Raake, "Evaluating QoE by means of traditional and alternative subjective measures: An exploratory'living room lab'study on IPTV," *Vienna, Austria*, 2013.

[85] I. Orsolic, M. Suznjevic, and L. Skorin-Kapov, "YouTube QoE estimation from encrypted traffic: Comparison of test methodologies and machine learning based models," in *2018 Tenth International Conference on Quality of Multimedia Experience (QoMEX)*, IEEE, 2018, pp. 1–6.

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

[87] S. Möller, *Assessment and prediction of speech quality in telecommunications*. Springer, 2000, vol. ISBN: 0792378946.

[88] M. Fiedler, S. Möller, and P. Reichl, "Quality of experience: From user perception to instrumental metrics (dagstuhl seminar 12181)," *Dagstuhl Reports*, vol. 2, no. 5, 2012.

[89] ITU-T P.1201, *Parametric non-intrusive assessment of audiovisual media streaming quality. Amendment 2: New Appendix III – Use of ITU-T P.1201 for non-adaptive, progressive download type media streaming*, International Telecommunications Union, Dec. 2013.

[90] A Richards, P. Rogers, V Witana, and M Antoniades, "Mapping user level QoS from a single parameter," in *International Conference on Multimedia Networks and Services (MMNS '98)*, Versailles, France, 1998.

[91] Z. G. Zhang and N. Tian, "The N threshold policy for the GI/M/1 queue," *Operations Research Letters*, vol. 32, no. 1, pp. 77–84, 2004.

[92] H. Riiser, P. Vigmostad, C. Griwodz, and P. Halvorsen, "Commute path bandwidth traces from 3G networks: Analysis and applications," in *Proceedings of the 4th ACM Multimedia Systems Conference*, ACM, 2013, pp. 114–118.

[93] B. A. Swett, J. L. Contreras-Vidal, R. Birn, and A. Braun, "Neural substrates of graphomotor sequence learning: A combined fMRI and kinematic study," *Journal of neurophysiology*, vol. 103, no. 6, pp. 3366–3377, 2010.

[94] K. Gakis, H.-K. Rhee, and B. Sivazlian, "Distributions and first moments of the busy and idle periods in controllable M/G/1 queueing models with simple and dyadic policies," *Stochastic Analysis and Applications*, vol. 13, no. 1, pp. 47–81, 1995.

[95] R. Schatz, T. Hoßfeld, L. Janowski, and S. Egger, "From packets to people: Quality of experience as a new measurement challenge," in *Data traffic monitoring and analysis*, Springer, 2013, pp. 219–263.

[96] C. Müller, S. Lederer, and C. Timmerer, "An evaluation of dynamic adaptive streaming over HTTP in vehicular environments," in *4th Workshop on Mobile Video*, ACM, 2012, pp. 37–42.

[97] T. Hoßfeld, M. Seufert, C. Sieber, T. Zinner, and P. Tran-Gia, "Identifying QoE optimal adaptation of HTTP adaptive streaming based on subjective studies," *Computer Networks*, vol. 81, pp. 320–332, 2015.

[98] P. Casas, A. Sackl, S. Egger, and R. Schatz, "YouTube & facebook quality of experience in mobile broadband networks," in *2012 IEEE Globecom Workshops*, IEEE, 2012, pp. 1269–1274.

[99] S. Egger, B. Gardlo, M. Seufert, and R. Schatz, "The impact of adaptation strategies on perceived quality of HTTP adaptive streaming," in *Proceedings of the 2014 Workshop on Design, Quality and Deployment of Adaptive Video Streaming*, ACM, 2014, pp. 31–36.

[100] J. W. Kleinrouweler, S. Cabrero, R. van der Mei, and P. Cesar, "A model for evaluating sharing policies for network-assisted HTTP adaptive streaming," *Computer Networks*, vol. 109, pp. 234–245, 2016.

[101] Y. Xu, Z. Xiao, H. Feng, T. Yang, B. Hu, and Y. Zhou, "Modeling buffer starvations of video streaming in cellular networks with large-scale measurement of user behavior," *IEEE Transactions on Mobile Computing*, 2016.

[102] 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 Transactions on Multimedia Computing, Communications, and Applications (TOMM)*, 2018, preprint on webpage goo.gl/UB6XL8.

[103] V. Joseph, S. Borst, and M. I. Reiman, "Optimal rate allocation for video streaming in wireless networks with user dynamics," *IEEE/ACM Transactions on Networking (TON)*, vol. 24, no. 2, pp. 820–835, 2016.

[104] S. Tanwir and H. Perros, "Modeling live adaptive streaming over HTTP," *Computer Communications*, vol. 85, pp. 74–88, 2016.

[105] Z. Ye, E.-A. Rachid, and T. Jimenez, "Analysis and modelling quality of experience of video streaming under time-varying bandwidth," in *Wireless and Mobile Networking Conference (WMNC), 2016 9th IFIP*, IEEE, 2016, pp. 145–152.

[106] X. Yin, A. Jindal, V. Sekar, and B. Sinopoli, "A control-theoretic approach for dynamic adaptive video streaming over HTTP," *ACM SIGCOMM Computer Communication Review*, vol. 45, no. 4, pp. 325–338, 2015.

[107] L. De Cicco, G. Cofano, and S. Mascolo, "A hybrid model of the akamai adaptive streaming control system," *Nonlinear Analysis: Hybrid Systems*, vol. 21, pp. 139–154, 2016.

[108] G. Tian and Y. Liu, "Towards agile and smooth video adaptation in dynamic HTTP streaming," in *Proceedings of the 8th international conference on Emerging networking experiments and technologies*, ACM, 2012, pp. 109–120.

[109] Y. Xu, S. E. Elayoubi, E. Altman, R. El-Azouzi, and Y. Yu, "Flow-level QoE of video streaming in wireless networks," *IEEE Transactions on Mobile Computing*, vol. 15, no. 11, pp. 2762–2780, 2016.

[110] Y. Xu, S. E. Elayoubi, E. Altman, and R. El-Azouzi, "Impact of flow-level dynamics on QoE of video streaming in wireless networks," in *INFOCOM, 2013 Proceedings IEEE*, IEEE, 2013, pp. 2715–2723.

[111] Y. Xu, Y. Zhou, and D.-M. Chiu, "Analytical QoE models for bit-rate switching in dynamic adaptive streaming systems," *IEEE Transactions on Mobile Computing*, vol. 13, no. 12, pp. 2734–2748, 2014.

[112] T. Bonald, S. E. Elayoubi, and Y.-T. Lin, "A flow-level performance model for mobile networks carrying adaptive streaming traffic," in *Global Communications Conference (GLOBECOM), 2015 IEEE*, IEEE, 2015, pp. 1–7.

[113] Y.-T. Lin, T. Bonald, and S. E. Elayoubi, "Impact of chunk duration on adaptive streaming performance in mobile networks," in *Wireless Communications and Networking Conference (WCNC), 2016 IEEE*, IEEE, 2016, pp. 1–6.

[114] K. Miller, N. Corda, S. Argyropoulos, A. Raake, and A. Wolisz, "Optimal adaptation trajectories for block-request adaptive video streaming," in *2013 20th International Packet Video Workshop*, IEEE, 2013, pp. 1–8.

[115] T. Hoßfeld, M. Seufert, C. Sieber, and T. Zinner, "Assessing effect sizes of influence factors towards a QoE model for http adaptive streaming," in *2014 sixth international workshop on quality of multimedia experience (qomex)*, IEEE, 2014, pp. 111–116.

[116] H. Nam, K.-H. Kim, and H. Schulzrinne, "QoE matters more than QoS: Why people stop watching cat videos," in *IEEE INFOCOM 2016-The 35th Annual IEEE International Conference on Computer Communications*, IEEE, 2016, pp. 1–9.

[117] L. Yao, W. Qi, G. Lei, S. Bo, C. Songqing, and L. Yingjie, "Investigating redundant internet video streaming traffic on iOS devices: Causes and solutions," *Multimedia, IEEE Transactions on*, vol. 16, no. 2, 2014.

[118] A. Rao, A. Legout, Y.-s. Lim, D. Towsley, C. Barakat, and W. Dabbous, "Network characteristics of video streaming traffic," in *Proceedings of the Seventh COnference on emerging Networking EXperiments and Technologies*, ACM, 2011.

[119] M. Ito, R. Antonello, D. Sadok, and S. Fernandes, "Network level characterization of adaptive streaming over HTTP applications," in *IEEE Symposium on Computers and Communication (ISCC)*, 2014. DOI: 10.1109/ISCC.2014.6912603.

[120] J. Añorga, S. Arrizabalaga, B. Sedano, M. Alonso-arce, and J. Mendizabal, "YouTube's DASH implementation analysis," in *19th International Conference on Circuits, Systems, Communications and Computers (CSCC)*, 2015, pp. 61–66, ISBN: 9781618043184.

[121] S. Alcock and R. Nelson, "Application flow control in YouTube video streams," *SIGCOMM Comput. Commun. Rev.*, vol. 41, no. 2, Apr. 2011. DOI: 10.1145/1971162.1971166.

[122] 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, MoViD'14*, 2014, ISBN: 9781450327077. DOI: 10.1145/2579465.2579469.

[123] Y. Liu, F. Li, L. Guo, B. Shen, and S. Chen, "A comparative study of android and iOS for accessing internet streaming services," in *Passive and Active Measurement*, Springer, 2013, pp. 104–114.

[124] H. Nam, B. H. Kim, D. Calin, and H. G. Schulzrinne, "Mobile video is inefficient: A traffic analysis," 2013.

[125] C. Sieber, P. E. Heegaard, T. Hoßfeld, and W. Kellerer, "Sacrificing efficiency for quality of experience: YouTube's redundant traffic behavior," in *IFIP Networking 2016 Conference (Networking 2016)*, Vienna, Austria, May 2016.

[126] I. 23009-5:2017, "Information technology - dynamic adaptive streaming over HTTP (DASH) - part 5: Server and network assisted DASH (SAND)," International Organization for Standardization, Geneva, Switzerland, ISO, 2017.

[127] I. Ben Mustafa, T. Nadeem, and E. Halepovic, "Flexstream: Towards flexible adaptive video streaming on end devices using extreme sdn," in *2018 ACM Multimedia Conference on Multimedia Conference*, ACM, 2018, pp. 555–563.

[128] A. Bentaleb, A. C. Begen, S. Harous, and R. Zimmermann, "A distributed approach for bitrate selection in HTTP adaptive streaming," in *2018 ACM Multimedia Conference on Multimedia Conference*, ACM, 2018, pp. 573–581.

[129] S. Akhshabi, L. Anantakrishnan, A. C. Begen, and C. Dovrolis, "What happens when HTTP adaptive streaming players compete for bandwidth?" In *Proceedings of the 22nd international workshop on Network and Operating System Support for Digital Audio and Video*, ACM, 2012, pp. 9–14.

[130] I. Ayad, Y. Im, E. Keller, and S. Ha, "A practical evaluation of rate adaptation algorithms in http-based adaptive streaming," *Computer Networks*, vol. 133, pp. 90–103, 2018.

[131] 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*, ACM, 2013, pp. 19–24.

[132] P. Georgopoulos, Y. Elkhatib, M. Broadbent, M. Mu, and N. Race, "Towards network-wide QoE fairness using openflow-assisted adaptive video streaming," in *Proceedings of the 2013 ACM SIGCOMM workshop on Future human-centric multimedia networking*, ACM, 2013, pp. 15–20.

[133] A. M. Kakhki, S. Jero, D. Choffnes, C. Nita-Rotaru, and A. Mislove, "Taking a long look at QUIC: An approach for rigorous evaluation of rapidly evolving transport protocols," in *Proceedings of the 2017 Internet Measurement Conference*, ACM, 2017, pp. 290–303.

[134] D. Bhat, A. Rizk, and M. Zink, "Not so QUIC: A performance study of DASH over QUIC," in *Proceedings of the 27th Workshop on Network and Operating Systems Support for Digital Audio and Video*, ACM, 2017, pp. 13–18.

[135] R. K. Jain, D.-M. W. Chiu, and W. R. Hawe, "A quantitative measure of fairness and discrimination," *Eastern Research Laboratory, Digital Equipment Corporation, Hudson, MA*, 1984.

[136] T. Hoßfeld, L. Skorin-Kapov, P. E. Heegaard, and M. Varela, "Definition of QoE fairness in shared systems," *IEEE Communications Letters*, vol. 21, no. 1, pp. 184–187, 2016.

[137] P. L. Weiss, D. Rand, N. Katz, and R. Kizony, "Video capture virtual reality as a flexible and effective rehabilitation tool," *Journal of neuroengineering and rehabilitation*, vol. 1, no. 1, p. 12, 2004.

[138] M. Hosseini and V. Swaminathan, "Adaptive 360 VR video streaming: Divide and conquer," in *Multimedia (ISM), 2016 IEEE International Symposium on*, IEEE, 2016, pp. 107–110.

[139] 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*, ACM, 2016, pp. 1–6.

[140] M. Graf, C. Timmerer, and C. Mueller, "Towards bandwidth efficient adaptive streaming of omnidirectional video over HTTP: Design, implementation, and evaluation," in *Proceedings of the 8th ACM on Multimedia Systems Conference*, ACM, 2017, pp. 261–271.

[141] R. I. T. da Costa Filho, M. C. Luizelli, M. T. Vega, J. van der Hooft, S. Petrangeli, T. Wauters, F. De Turck, and L. P. Gaspary, "Predicting the performance of virtual reality video streaming in mobile networks," in *Proceedings of the 9th ACM Multimedia Systems Conference*, ACM, 2018, pp. 270–283.

[142] S. Petrangeli, V. Swaminathan, M. Hosseini, and F. De Turck, "An HTTP/2-based adaptive streaming framework for 360 virtual reality videos," in *Proceedings of the 25th ACM international conference on Multimedia*, ACM, 2017, pp. 306–314.

[143] C. Zhou, Z. Li, and Y. Liu, "A measurement study of oculus 360 degree video streaming," in *Proceedings of the 8th ACM on Multimedia Systems Conference*, ACM, 2017, pp. 27–37.

[144] M. B. 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)*, IEEE, 2018, pp. 89–96.

[145] S. Petrangeli, G. Simon, and V. Swaminathan, "Trajectory-based viewport prediction for 360-degree virtual reality videos," in *2018 IEEE International Conference on Artificial Intelligence and Virtual Reality (AIVR)*, IEEE, 2018, pp. 157–160.

[146] C. Li, W. Zhang, Y. Liu, and Y. Wang, "Very long term field of view prediction for 360-degree video streaming," *arXiv preprint arXiv:1902.01439*, 2019.

[147] J. van der Hooft, M. T. Vega, S. Petrangeli, T. Wauters, and F. De Turck, "Optimizing adaptive tile-based virtual reality video streaming," in *2019 IFIP/IEEE Symposium on Integrated Network and Service Management (IM)*, IEEE, 2019, pp. 381–387.

[148] X. Corbillon, G. Simon, A. Devlic, and J. Chakareski, "Viewport-adaptive navigable 360-degree video delivery," in *2017 IEEE international conference on communications (ICC)*, IEEE, 2017, pp. 1–7.

[149] X. Corbillon, F. De Simone, and G. Simon, "360-degree video head movement dataset," in *Proceedings of the 8th ACM on Multimedia Systems Conference*, ACM, 2017, pp. 199–204.

[150] L. Xie, Z. Xu, Y. Ban, X. Zhang, and Z. Guo, "360probdash: Improving QoE of 360 video streaming using tile-based HTTP adaptive streaming," in *Proceedings of the 25th ACM international conference on Multimedia*, ACM, 2017, pp. 315–323.

[151] C.-L. Fan, J. Lee, W.-C. Lo, C.-Y. Huang, K.-T. Chen, and C.-H. Hsu, "Fixation prediction for 360 video streaming in head-mounted virtual reality," in *Proceedings of the 27th Workshop on Network and Operating Systems Support for Digital Audio and Video*, ACM, 2017, pp. 67–72.

[152] Q. Yang, J. Zou, K. Tang, C. Li, and H. Xiong, "Single and sequential viewports prediction for 360-degree video streaming," in *2019 IEEE International Symposium on Circuits and Systems (ISCAS)*, IEEE, 2019, pp. 1–5.

[153] J. Heyse, M. T. Vega, F. De Backere, and F. De Turck, "Contextual bandit learning-based viewport prediction for 360 video," *IEEE Virtual Reality (VR)*, 2019.

[154] X. Feng, V. Swaminathan, and S. Wei, "Viewport prediction for live 360-degree mobile video streaming using user-content hybrid motion tracking," *Proceedings of the ACM on Interactive, Mobile, Wearable and Ubiquitous Technologies*, vol. 3, no. 2, p. 43, 2019.

[155] L. Sun, F. Duanmu, Y. Liu, Y. Wang, Y. Ye, H. Shi, and D. Dai, "A two-tier system for on-demand streaming of 360 degree video over dynamic networks," *IEEE Journal on Emerging and Selected Topics in Circuits and Systems*, vol. 9, no. 1, pp. 43–57, 2019.

[156] C. Sieber, P. Heegaard, T. Hoßfeld, and W. Kellerer, "Sacrificing efficiency for quality of experience: YouTube's redundant traffic behavior," in *IFIP Networking Conference (IFIP Networking) and Workshops, 2016*, IEEE, 2016, pp. 503–511.

[157] C. Sieber, A. Blenk, M. Hinteregger, and W. Kellerer, "The cost of aggressive HTTP adaptive streaming: Quantifying YouTube's redundant traffic," in *IFIP/IEEE International Symposium on Integrated Network Management (IM) 2015*, May 2015, pp. 1261–1267. DOI: 10.1109/INM.2015.7140478.

[158] *Experimental video dataset.* [Online]. Available: https://git.io/vRSSW.

[159] R. E. Barlow, D. J. Bartholomew, J. Bremner, and H. D. Brunk, *Statistical inference under order restrictions: the theory and application of isotonic regression*. Wiley New York, 1972.

[160] Y. Liu, S. Dey, D. Gillies, F. Ulupinar, and M. Luby, "User experience modeling for DASH video," in *20th Int, Packet Video Workshop (PV)*, IEEE, 2013, pp. 1–8.

[161] J. Chen, M. Ammar, M. Fayed, and R. Fonseca, "Client-driven network-level QoE fairness for encrypted'DASH-S'," in *Proceedings of the 2016 workshop on QoE-based Analysis and Management of Data Communication Networks*, ACM, 2016, pp. 55–60.

[162] A. Mansy, M. Fayed, and M. Ammar, "Network-layer fairness for adaptive video streams," in *2015 IFIP Networking Conference (IFIP Networking)*, IEEE, 2015, pp. 1–9.

[163] T. Hoßfeld, P. E. Heegaard, L. Skorin-Kapov, and M. Varela, "No silver bullet: QoE metrics, QoE fairness, and user diversity in the context of QoE management," in *2017 Ninth International Conference on Quality of Multimedia Experience (QoMEX)*, IEEE, 2017, pp. 1–6.

[164] 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*, ACM, 2016, p. 3.

[165] A. Finamore, M. Mellia, M. M. Munafò, R. Torres, and S. G. Rao, "YouTube everywhere: Impact of device and infrastructure synergies on user experience," in *Proceedings of the 2011 ACM SIGCOMM conference on Internet measurement conference*, ACM, 2011, pp. 345–360.

[166] C. Mueller, S. Lederer, R. Grandl, and C. Timmerer, "Oscillation compensating dynamic adaptive streaming over HTTP," in *2015 IEEE International Conference on Multimedia and Expo (ICME)*, IEEE, 2015, pp. 1–6.

[167] S. Petrangeli, T. Wauters, R. Huysegems, T. Bostoen, and F. De Turck, "Network-based dynamic prioritization of HTTP adaptive streams to avoid video freezes," in *2015 IFIP/IEEE International Symposium on Integrated Network Management (IM)*, IEEE, 2015, pp. 1242–1248.

[168] K. Khan and W. Goodridge, "Server-based and network-assisted solutions for adaptive video streaming," *International Journal of Advanced Networking and Applications*, vol. 9, no. 3, pp. 3432–3442, 2017.

[169] J. Van Der Hooft, S. Petrangeli, T. Wauters, R. Huysegems, P. R. Alface, T. Bostoen, and F. De Turck, "HTTP/2-based adaptive streaming of HEVC video over 4G/LTE networks," *IEEE Communications Letters*, vol. 20, no. 11, pp. 2177–2180, 2016.

[170] *Conviva 2019 state of streaming*, 2019. [Online]. Available: https://www.conviva.com/research/convivas-state-streaming-tv-industry-q2-2019/ (visited on 07/12/2020).

[171] J. Janek and W. G. Zeier, "A solid future for battery development," *Energy*, vol. 500, no. 400, p. 300, 2016.

[172] M. Siekkinen, M. A. Hoque, and J. K. Nurminen, "Using viewing statistics to control energy and traffic overhead in mobile video streaming," *IEEE/ACM Transactions on Networking*, vol. 24, no. 3, pp. 1489–1503, 2016.

[173] Cisco, "Cisco annual internet report (2018–2023) white paper," Tech. Rep., 2020. [Online]. Available: https://www.cisco.com/c/en/us/solutions/collateral/executive-perspectives/annual-internet-report/white-paper-c11-741490.html (visited on 07/12/2020).

[174] M. Seufert, N. Wehner, F. Wamser, P. Casas, A. D'Alconzo, and P. Tran-Gia, "Unsupervised QoE field study for mobile YouTube video streaming with yomoapp," in *2017 Ninth International Conference on Quality of Multimedia Experience (QoMEX)*, IEEE, 2017, pp. 1–6.

[175] H Riiser, P Vigmostad, C Griwodz, and P Halvorsen, *Dataset: HSDPA-bandwidth logs for mobile HTTP streaming scenarios*, 2012.

[176] H. Yu, D. Zheng, B. Y. Zhao, and W. Zheng, "Understanding user behavior in large-scale video-on-demand systems," in *ACM SIGOPS Operating Systems Review*, ACM, vol. 40, 2006, pp. 333–344.

[177] F. Dobrian, A. Awan, D. Joseph, A. Ganjam, J. Zhan, V. Sekar, I. Stoica, and H. Zhang, "Understanding the impact of video quality on user engagement," *Communications of the ACM*, vol. 56, no. 3, pp. 91–99, 2013.

[178] L. A. Liikkanen and A. Salovaara, "Music on YouTube: User engagement with traditional, user-appropriated and derivative videos," *Computers in Human Behavior*, vol. 50, pp. 108–124, 2015.

[179] A. McGowan, P. Hanna, and N. Anderson, "Teaching programming: Understanding lecture capture YouTube analytics," in *Proceedings of the 2016 ACM Conference on Innovation and Technology in Computer Science Education*, ACM, 2016, pp. 35–40.

[180] M. Plakia, E. Tzamousis, T. Asvestopoulou, G. Pantermakis, N. Filippakis, H. Schulzrinne, Y. Kane-Esrig, and M. Papadopouli, "Should i stay or should i go: Analysis of the impact of application QoS on user engagement in YouTube," *arXiv preprint arXiv:1901.01603*, 2019.

[181] P. Reichl, S. Egger, S. Möller, 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)*, IEEE, 2015, pp. 1–6.

[182] L. Chen, Y. Zhou, and D. M. Chiu, "Video browsing-a study of user behavior in online vod services," in *2013 22nd International Conference on Computer Communication and Networks (ICCCN)*, IEEE, 2013, pp. 1–7.

[183] V. Gopalakrishnan, R. Jana, K. Ramakrishnan, D. F. Swayne, and V. A. Vaishampayan, "Understanding couch potatoes: Measurement and modeling of interactive usage of IPTV at large scale," in *Proceedings of the 2011 ACM SIGCOMM conference on Internet measurement conference*, ACM, 2011, pp. 225–242.

[184] L. Mikos, "Digital media platforms and the use of TV content: Binge watching and video-on-demand in germany," *Media and Communication*, vol. 4, no. 3, pp. 154–161, 2016.

[185] X. Li, Z. Ma, F. C. Fernandes, *et al.*, "Modeling power consumption for video decoding on mobile platform and its application to power-rate constrained streaming," in *Visual Communications and Image Processing (VCIP), 2012 IEEE*, IEEE, 2012, pp. 1–6.

[186] T. GPP, "3G radio resource control; protocol specification; 3GPP technical specification 25.331," 2020.

[187] H. Haverinen, J. Siren, and P. Eronen, "Energy consumption of always-on applications in WCDMA networks," in *2007 IEEE 65th Vehicular Technology Conference-VTC2007-Spring*, IEEE, 2007, pp. 964–968.

[188] T. GPP, *LTE radio resource control; protocol specification; 3GPP technical specification 36.331*, 2020.

[189] D. Vinella and M. Polignano, "Discontinuous reception and transmission (DRX/DTX) strategies in long term evolution (LTE) for Voice-Over-IP (VOIP) traffic under both full-dynamic and semi-persistent packet scheduling policies," *Project Group*, vol. 996, 2009.

[190] F. Qian, Z. Wang, A. Gerber, Z. M. Mao, S. Sen, and O. Spatscheck, "Characterizing radio resource allocation for 3G networks," in *Proceedings of the 10th ACM SIGCOMM conference on Internet measurement*, ACM, 2010, pp. 137–150.

[191] F. Qian, Z. Wang, A. Gerber, Z. M. Mao, S. Sen, and O. Spatscheck, "Top: Tail optimization protocol for cellular radio resource allocation," in *Network Protocols (ICNP), 2010 18th IEEE International Conference on*, IEEE, 2010, pp. 285–294.

[192] M. A. Hoque, M. Siekkinen, J. K. Nurminen, and M. Aalto, "Dissecting mobile video services: An energy consumption perspective," in *World of Wireless, Mobile and Multimedia Networks (WoWMoM), 2013 IEEE 14th International Symposium and Workshops on a*, IEEE, 2013, pp. 1–11.

[193] M. Seufert, V. Burger, and F. Kaup, "Evaluating the impact of WiFi offloading on mobile users of HTTP adaptive video streaming," in *Globecom Workshops (GC Wkshps), 2016 IEEE*, IEEE, 2016, pp. 1–6.

[194] C. Schwartz, M. Scheib, T. Hoßfeld, P. Tran-Gia, and J. M. Gimenez-Guzman, "Trade-offs for video-providers in LTE networks: Smartphone energy consumption vs wasted traffic," in *Energy Efficient and Green Networking (SSEEGN), 2013 22nd ITC Specialist Seminar on*, IEEE, 2013, pp. 1–6.

[195] M. Siekkinen, M. A. Hoque, J. K. Nurminen, and M. Aalto, "Streaming over 3G and LTE: How to save smartphone energy in radio access network-friendly way," in *Proceedings of the 5th Workshop on Mobile Video*, ACM, 2013, pp. 13–18.

[196] *Alexa*, 2016. [Online]. Available: http://www.alexa.com/topsites/countries/DE (visited on 11/03/2016).

[197] J. Aas, *Progress towards 100% HTTPS, june 2016*, 2016. [Online]. Available: https://letsencrypt.org/2016/06/22/https-progress-june-2016.html (visited on 07/12/2020).

[198] Google, "Google transparency report - HTTPS encryption on the web," 2020. [Online]. Available: https://transparencyreport.google.com/https/overview?hl=en (visited on 07/12/2020).

[199] D. F. Galletta, R. Henry, S. McCoy, and P. Polak, "Web site delays: How tolerant are users?" *Journal of the Association for Information Systems*, vol. 5, no. 1, p. 1, 2004.

[200] F. Wamser, M. Seufert, P. Casas, R. Irmer, P. Tran-Gia, and R. Schatz, "Yomoapp: A tool for analyzing QoE of YouTube HTTP adaptive streaming in mobile networks," in *2015 European Conference on Networks and Communications (EuCNC)*, IEEE, 2015, pp. 239–243.

[201] S. Ickin, K. Wac, M. Fiedler, L. Janowski, J.-H. Hong, and A. K. Dey, "Factors influencing quality of experience of commonly used mobile applications," *IEEE Communications Magazine*, vol. 50, no. 4, pp. 48–56, 2012.

[202] M. Siekkinen, M. A. Hoque, and J. K. Nurminen, "Using viewing statistics to control energy and traffic overhead in mobile video streaming," *IEEE/ACM Transactions on Networking*, vol. 24, no. 3, pp. 1489–1503, 2015.

[203] H. Mao, R. Netravali, and M. Alizadeh, "Neural adaptive video streaming with pensieve," in *Proceedings of the Conference of the ACM Special Interest Group on Data Communication*, ACM, 2017, pp. 197–210.

[204] J. Huang, F. Qian, A. Gerber, Z. M. Mao, S. Sen, and O. Spatscheck, "A close examination of performance and power characteristics of 4G LTE networks," in *Proceedings of the 10th international conference on Mobile systems, applications, and services*, ACM, 2012, pp. 225–238.

[205] F. Dormancy, "Fast dormancy best practices," *GSM association, network efficiency task force*, 2010.

[206] S. Petrangeli, J. Famaey, M. Claeys, S. Latré, and F. De Turck, "QoE-driven rate adaptation heuristic for fair adaptive video streaming," *ACM Transactions on Multimedia Computing, Communications, and Applications (TOMM)*, vol. 12, no. 2, p. 28, 2016.

[207] A. Finamore, M. Mellia, M. M. Munafò, R. Torres, and S. G. Rao, "YouTube everywhere: Impact of device and infrastructure synergies on user experience," in *Proceedings of the 2011 ACM SIGCOMM conference on Internet measurement conference*, ACM, 2011, pp. 345–360.

ISSN 1432-8801