QoE-Based Low-Delay Live Streaming Using Throughput Predictions

KONSTANTIN MILLER, Technische Universität Berlin
ABDEL-KARIM AL-TAMIMI, Yarmouk University
ADAM WOLISZ, Technische Universität Berlin

Recently, Hypertext Transfer Protocol (HTTP)-based adaptive streaming has become the de facto standard for video streaming over the Internet. It allows clients to dynamically adapt media characteristics to the varying network conditions to ensure a high quality of experience (QoE)—that is, minimize playback interruptions while maximizing video quality at a reasonable level of quality changes. In the case of live streaming, this task becomes particularly challenging due to the latency constraints. The challenge further increases if a client uses a wireless access network, where the throughput is subject to considerable fluctuations. Consequently, live streams often exhibit latencies of up to 20 to 30 seconds. In the present work, we introduce an adaptation algorithm for HTTP-based live streaming called *LOLYPOP* (short for low-latency prediction-based adaptation), which is designed to operate with a transport latency of a few seconds. To reach this goal, LOLYPOP leverages Transmission Control Protocol throughput predictions on multiple time scales, from 1 to 10 seconds, along with estimations of the relative prediction error distributions. In addition to satisfying the latency constraint, the algorithm heuristically maximizes the QoE by maximizing the average video quality as a function of the number of skipped segments and quality transitions. To select an efficient prediction method, we studied the performance of several time series prediction methods in IEEE 802.11 wireless access networks. We evaluated LOLYPOP under a large set of experimental conditions, limiting the transport latency to 3 seconds, against a state-of-the-art adaptation algorithm called *FESTIVE*. We observed that the average selected video representation index is by up to a factor of 3 higher than with the baseline approach. We also observed that LOLYPOP is able to reach points from a broader region in the QoE space, and thus it is better adjustable to the user profile or service provider requirements.

CCS Concepts: • **Information systems** → **Multimedia streaming**; • **Networks** → *Network dynamics;*

Additional Key Words and Phrases: Adaptive streaming, MPEG-DASH, TCP throughput prediction

ACM Reference Format:
Konstantin Miller, Abdel-Karim Al-Tamimi, and Adam Wolisz. 2016. QoE-based low-delay live streaming using throughput predictions. ACM Trans. Multimedia Comput. Commun. Appl. 13, 1, Article 4 (October 2016), 24 pages.
DOI: http://dx.doi.org/10.1145/2990505

1. INTRODUCTION

Over the past few years, we have been observing a dramatic change in video consumption patterns. The era of passive consumption of noninteractive "linear" content on a single device, the TV set, appears to be coming to an end, and a new mind-set is being established: watch what I want, when I want, and where I want [ComScore 2014;

This work was supported by the ZIM programme of the German Federal Ministry for Economic Affairs and Energy (BMWi) in the context of the research project ANOPAS.
Authors' addresses: K. Miller and A. Wolisz, Fachg. TKN, Sekr. FT5, Einsteinufer 25, 10587 Berlin, Germany; emails: {konstantin.miller, adam.wolisz}@tu-berlin.de; A.-K. Al-Tamimi, Computer engineering department, Yarmouk university, Irbid, 21163, Jordan; email: altamimi@yu.edu.jo.
Permission to make digital or hard copies of part or all of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies show this notice on the first page or initial screen of a display along with the full citation. Copyrights for components of this work owned by others than ACM must be honored. Abstracting with credit is permitted. To copy otherwise, to republish, to post on servers, to redistribute to lists, or to use any component of this work in other works requires prior specific permission and/or a fee. Permissions may be requested from Publications Dept., ACM, Inc., 2 Penn Plaza, Suite 701, New York, NY 10121-0701 USA, fax +1 (212) 869-0481, or permissions@acm.org.
© 2016 ACM 1551-6857/2016/10-ART4 $15.00
DOI: http://dx.doi.org/10.1145/2990505

Conviva 2015]. This transformation is accompanied by a significant increase in wireless and mobile network usage. In 2013, wired devices still accounted for the majority of Internet traffic at 56%. However, wireless and mobile device traffic is predicted to exceed traffic from wired devices by 2018, accounting for 61% of the total Internet traffic. The largest part of the transmitted data will be video content [Cisco 2014].

Although most of the streamed content is video on demand (VoD), the amount of live streaming is growing rapidly [Sweeting 2014]. Whereas current live streaming services may exhibit a latency of several tens of seconds, low-delay streaming refers to live streaming with a particularly low upper bound on the latency: a few seconds or less. Such a requirement is desirable for scenarios such as transmissions of sports events. Moreover, it is absolutely necessary in the case of video conferencing and online gaming, where active participants have latency requirements on the order of hundreds of milliseconds [ITU-T 2014], whereas permanently or temporarily passive participants may be served with a delay of a few seconds.

Streaming video over the Internet has always been a challenging task because the Internet was not designed to support applications that require guaranteed end-to-end quality of service (QoS) [ITU-T 2008a]. Even though considerable effort has been put into developing networking architectures addressing this shortcoming [Aurrecoechea et al. 1998; Carapinha et al. 2010], none of the approaches have achieved a significant pervasiveness. As a result, in 2013, around 26.9% of streaming sessions on the Internet experienced playback interruption, 43.3% were impacted by low resolution, and 4.8% failed to even start altogether [Conviva 2014]. Especially on wireless links, users are exposed to interference, cross-traffic, and fading effects, leading to continuously fluctuating QoS characteristics.

As a consequence, we lately have been observing a period of high interest in adaptive streaming technologies that are able to dynamically adjust the characteristics of the streamed media to the varying network conditions, leading to a smoother viewing experience with less playback interruptions and a more efficient utilization of the available network resources. In particular, one technology has become the de facto standard for over-the-top (OTT) streaming: HTTP-based adaptive streaming (HAS) [Stockhammer 2011]. One of the enablers of its success was the open standard MPEG-DASH (dynamic adaptive streaming over HTTP) [MPEG 2012; Sodagar 2011]. The advantage of HAS comes from its usage of the HTTP leveraging an ubiquitous and highly optimized delivery infrastructure, including content delivery networks (CDNs), caches, and proxies, and reducing the operating costs due to the lack of necessity to maintain specialized video servers and pay for their licenses. In addition, HTTP is typically allowed to traverse middleboxes, such as network address translation (NAT) devices and firewalls. Finally, HAS has good scalability properties due to the stateless nature of HTTP and considering that the control logic resides at the client.

HAS, however, was primarily developed to replace the progressive download of VoD content, and therefore its usage for low-delay streaming has received little attention in the research community. Typical buffer sizes used in studies for evaluation and deployment of HAS-based clients are on the order of tens of seconds. The capability of the HAS approach to efficiently stream low-delay content, especially in wireless networks, is still an open question.

One of the main goals of a streaming client's adaptation logic is to maximize quality of experience (QoE). The notion of QoE has been introduced in an effort to make the various phenomena affecting human perception of multimedia content accessible to an objective evaluation process [ITU-T 2008b]. Among the QoE influencing factors that are controlled by the adaptation logic are the number of playback interruptions, the number of quality transitions, and the video quality. It is worth noting that these three factors cannot be considered separately. For example, always selecting the lowest video quality minimizes playback interruptions and quality transitions and allows for the lowest playback latency. On the other hand, always selecting the highest video quality typically results in an unacceptably high number of playback interruptions.

In this study, we demonstrate that efficient HAS-based low-delay live streaming is possible by leveraging short-term Transmission Control Protocol (TCP) throughput predictions over multiple time scales, from 1 to 10 seconds, along with estimations of the relative prediction error distribution. We design a novel prediction-based algorithm called *LOLYPOP* (short for low-latency prediction-based adaptation), which supports QoE-based adaptation with a transport latency on the order of a few seconds. The approach introduced in LOLYPOP jointly considers four QoE components: latency, number of playback interruptions, number of quality transitions, and average video quality. Its goal is to maximize the average video quality as a function of the operating point defined by the other three components. The operating point is controlled by three input parameters: the target latency, an upper bound on the number of quality transitions, and a parameter controlling the number of playback interruptions. Thus, LOLYPOP provides configurable QoE that can be adjusted to the nature of the video, the user context and preferences, or the service provider's business model.

At the core of LOLYPOP is an estimation of the download success probabilities for the individual segments. To obtain the estimations, LOLYPOP leverages predictions of throughput distributions, computed from time series predictions and error estimations. We evaluate several time series prediction methods using TCP throughput traces collected in IEEE 802.11 wireless local area networks (WLANs), including public hotspots (indoor and outdoor), campus hotspots, and access points in residential environments. We observe, somewhat surprisingly, that taking the average over the previous T seconds as a prediction for the next T seconds provides the best prediction accuracy among the considered methods for all considered time scales. In other words, taking into account the trend does not help to reduce the prediction error.

We implement a prototype of the algorithm and evaluate it against FESTIVE [Jiang et al. 2014], a well-known adaptation algorithm from the literature. We limit the transport latency to 3 seconds using a segment duration of 2 seconds. We observe that LOLYPOP is able to reach a broad range of operating points and thus can be flexibly adapted to the user profile or service provider requirements. Furthermore, we observe that at the individual operating points, LOLYPOP provides an average video quality, expressed by the selected video representation index, which is by up to a factor of 3 higher than what is achieved by the baseline approach.

The rest of the article is structured as follows. After reviewing the related work in Section 2, we introduce the proposed system model and notation in Section 3. Next, we introduce the design of the proposed adaptation algorithm in Section 4. After describing our collected set of throughput traces in Section 5, we use them to compare the performance of several prediction methods in Section 6, where we also present our approach to computing download success probabilities. We evaluate the performance of LOLYPOP in Section 7 and present our conclusions in Section 8.

2. RELATED WORK

In the past few years, adaptive streaming has been a very active research area. Even though the idea is not new [Chen et al. 1995; Kanakia et al. 1995], it has recently moved into focus as the number of streaming services has increased, as well as the number of content providers using adaptive streaming on a large scale. HTTP was considered as a candidate application-layer streaming protocol as early as 2002 [Carmel et al. 2002], but it was not until recently that it became the technology of choice for delivering video over the Internet [Swaminathan 2013; Li et al. 2013]. The adoption of the open standard

MPEG-DASH [MPEG 2012; Sodagar 2011] in 2011 has significantly contributed to the popularity of HAS.

There is a large number of recent studies focusing on adaptation algorithms for HAS that do not address the low-delay requirement. They typically consider playback buffer sizes of 10 to 30 seconds or more. Various approaches have been proposed that are based on control theory [Zhou et al. 2012; Zhu et al. 2013; Zhou et al. 2014; Yin et al. 2014; Miller et al. 2015], Markov decision processes [Jarnikov and Özçelebi 2011; Bokani et al. 2013], machine learning [Claeys et al. 2014], dynamic programming [Li et al. 2014a], data-driven techniques [Liu et al. 2012; Riiser et al. 2012; Balachandran et al. 2013; Hao et al. 2014], and various heuristics selecting the video quality based on the client's playback buffer level and average throughput [Liu et al. 2011; Mok et al. 2012; Miller et al. 2012; Li et al. 2014b; Jiang et al. 2014]. Several studies use cross-layer information to improve the performance [Abdallah and Mackenzie 2015] or jointly consider the problem of video quality selection and resource allocation on the lower layers of the protocol stack [Miller et al. 2015; Le et al. 2015a; Bethanabhotla et al. 2015]. Further studies propose to coordinate the quality selection process among clients sharing a bottleneck link to allow for a more fair and efficient resource usage [Essaili et al. 2013]. Finally, many popular streaming platforms such as YouTube[1] and Netflix[2] are using HAS for their VoD services, often deploying proprietary adaptation algorithms. Unfortunately, due to the lack of a standard evaluation methodology and performance metrics for HAS systems, it is difficult to compare results of individual studies.

The adaptation algorithm FESTIVE, proposed by Jiang et al. [2014], has the goal of heuristically optimizing three performance metrics: maximizing the utilization of the available network capacity, minimizing the number and magnitude of quality transitions, and minimizing application-layer unfairness between clients sharing a capacity bottleneck. FESTIVE uses the harmonic mean over a fixed number of segment downloads to estimate the available throughput. Based on this estimation, it selects a target video quality and initiates a convergence procedure, whose speed depends on the distance between the target and the current media bit rate, and the amount of quality transitions the client has already performed in the recent past. These two factors are considered jointly to resolve the trade-off between the utilization and the amount of quality fluctuations. In addition, FESTIVE randomizes the interrequest delays to avoid the dependency of the adaptation trajectories on the initial state. The authors evaluate FESTIVE against several commercial players, revealing its superior performance along all three considered performance metrics.

The number of studies that specifically target live or low-delay HAS is significantly smaller. A potential reason is the significantly smaller market size for OTT live streaming services [Sweeting 2014] and a popular opinion that streaming over HTTP/TCP is less suitable for applications requiring a low delay. Lohmar et al. [2011] compare the delay and communication overhead in HTTP-based live streaming with the one in Real-Time Transport Protocol (RTP)-based systems. They observe that the transport delay is by one segment duration larger with HTTP and that the communication overhead becomes substantial for subsecond segment durations (approximately 31 kbps for 1-second segments). Thang et al. [2014] evaluate several adaptation algorithms using buffer sizes from 6 to 20 seconds. The authors observe that the used set of representations can significantly affect the performance of the individual methods, resulting in a factor of up to 2.8 higher average video quality. They also observe that none of the selected five methods performs best in all cases but that the ranking depends on the throughput variability. However, the study was performed using only one throughput

[1]https://www.youtube.com.
[2]https://www.netflix.com.

trace. Wei and Swaminathan [2014] demonstrate that the server push feature introduced in HTTP/2 can be used to support low-delay streaming by allowing a reduction in the segment duration, and thus the lower bound on the achievable delay, without suffering from the superlinear increase in the number of requests observed with HTTP/1.1. However, the study does not analyze the protocol overhead caused by response headers and the reduction in video compression rate due to the decreased group of pictures (GOP) size. Le et al. [2015b] propose an adaptation algorithm for live streaming that compares potential adaptation trajectories for the next few segments and heuristically maximizes the video quality represented by the just noticeable difference (JND) metric. The algorithm is evaluated using a maximum buffer size of 10 seconds and compared against two baseline approaches. The evaluation shows that the algorithm exhibits lower average video quality than the maximum of the two baseline methods. However, it is able to reduce the average step size of a quality transition by 32% with respect to the minimum of the two baseline methods. The performed evaluation used only one throughput trace.

Several studies assume perfect information about future throughput to compute optimal adaptation trajectories that can be used to benchmark existing algorithms and evaluate the potential for performance increase [Miller et al. 2013; Zou et al. 2015]. The evaluation in Miller et al. [2013] reveals that the tested streaming clients can achieve a performance that is close to optimum with respect to the average media bit rate and number of playback interruptions, but with an average number of quality transitions that is at least several times as high as the optimum. The evaluated buffer sizes vary between 10 and 40 seconds.

There exist other studies that explicitly use throughput predictions in the context of video quality adaptation. Mok et al. [2012] consider path probing techniques to obtain throughput estimations that, however, typically require support from the network infrastructure, server instrumentation, and/or modifying lower protocol layers. The proposed adaptation algorithm is not evaluated. Similar in spirit to our work is the study by Liu and Lee [2014], which uses predictions to match a target number of skipped segments and a minimum delay between quality transitions to control the transitions frequency. The algorithm is evaluated in a mobile network and compared against two baseline approaches. The evaluation reveals that the baseline approaches, using default configurations, require up to an order of magnitude more quality transitions to achieve a similar media bit rate and a similar number of skipped segments as the proposed algorithm. Unfortunately, only one network environment is used for the evaluation, and the characteristics of the observed throughput dynamics are not disclosed in the study. In addition, the maximum buffer size used in the evaluation is not specified. Yin et al. [2014] study the effect of prediction errors on the performance of two adaptation approaches—rate based and model predictive control (MPC)—compared against a buffer-based approach using synthetic throughput traces. The authors conclude that when the prediction error exceeds 25%, prediction-based approaches might exhibit worse performance than the buffer-based algorithm. The used MPC algorithm is not described in the publication. Tian and Liu [2012] propose a prediction-based adaptation algorithm, where the media bit rate is selected to equal the predicted throughput times a dynamically varying adjustment factor. The proposed approach is, however, not designed to support QoE-based performance targets and has a set of configuration parameters that does not allow for a straightforward tuning of the algorithm for particular network environments. Li et al. [2014] use dynamic programming to solve a network utility maximization (NUM) problem for a finite time horizon, for which a bandwidth estimation is computed based on an exponentially weighted moving average (EWMA) of the recent segment downloads. The proposed adaptation algorithm is evaluated using buffer size limits between 30 and 50 seconds. Le et al. [2013] propose a heuristic adaptation algorithm that tries to satisfy constraints on future buffer levels over a finite time horizon using throughput predictions obtained using a modified EWMA model where the weights are dynamically adjusted based on the most recent relative prediction error. The algorithm is evaluated against two baseline approaches using a single throughput trace and a buffer size of 20 seconds. The evaluation reveals that the average video quality offered by the proposed algorithm is 3% (17%) higher, the average quality transition magnitude is 27% (9%) lower, and the number of quality transitions is 80% higher (46% lower) as compared to the two baseline approaches.

Since having accurate throughput predictions is beneficial for several applications, there are dedicated studies that address this subject (see He et al. [2007] for an overview). Since, however, the main application of TCP was for a long time bulk data transfer, many of those studies predict throughput averaged over much longer time intervals than required for low-delay streaming [Padhye et al. 2000; He et al. 2007]. He et al. [2007] observed that time series prediction methods perform quite well on the time scale of 50 seconds (root mean square relative error (RMSRE)) less than 0.4 for about 90% of studied traces), but the accuracy strongly varies across studied network paths. Moreover, evaluations are often limited to wired networks, which, however, are not subject to channel state dynamics and effects caused by the data link layer to the extent seen with wireless technologies [Padhye et al. 2000; He et al. 2007; Mirza et al. 2010]. Some of the developed approaches use information available only at the sender, or they require cross-layer information or additional path measurements that need to be supported by the other end point [Padhye et al. 2000; Mirza et al. 2010]. In addition, some analytical models target specific TCP flavors, making their performance uncertain given recently developed variants [Padhye et al. 2000].

QoE for adaptive video streaming is an important and fast-developing research area [Balachandran et al. 2012; Seufert et al. 2014; Reiter et al. 2014; Song and Tjondronegoro 2014]. The number of factors influencing QoE is immense, and many of them have a high level of subjectivity that results in extremely complex modeling [Reiter et al. 2014]. Rebuffering, initial delay, and quality fluctuations have not been part of traditional QoE metrics for video, but they have a tremendous impact on the user's perception of adaptive video streamed over a best-effort network such as the Internet. In particular, many studies suggest that the number and duration of playback interruptions have the most severe impact on QoE [Conviva 2014]. Users are willing to accept a higher initial delay and higher video distortion if it helps to minimize playback interruptions [Seufert et al. 2014; Hossfeld et al. 2012; Quan and Ghanbari 2008; Singh et al. 2012]. On the other hand, it was observed that video quality fluctuations resulting from dynamically changing the representation can have a negative impact on QoE [Lewcio et al. 2011; Yitong et al. 2013]. In particular, some studies conclude that a lower average video quality might be tolerated if it helps to reduce the amount of representation transitions [Pessemier et al. 2013]. User engagement is another important metric, which is especially of interest to content providers because it directly relates to advertising-based revenue schemes [Balachandran et al. 2013; Conviva 2014].

3. SYSTEM MODEL AND NOTATION

In this section, we outline the main operation principles of an HTTP-based adaptive low-delay live streaming system, along with the specific assumptions we have made for our study. Subsequently, we introduce the notation used throughout the article.

3.1. System Model

In a live streaming system, the video content is recorded and published while being streamed, in contrast to being prerecorded and stored at the server as in the case of VoD. The difference between the time when the content is recorded and the time when it is rendered on the user's device is often termed *live latency*. To provide the "live" experience, it is typically constrained by an upper bound. This severely limits the capability of the client to prefetch content to alleviate the transport delay variations caused by the varying network conditions, thus making the design of the system more challenging. The live latency consists of several components: sever-side processing (cutting, encoding, etc.), publishing (making available for download, distributing among CDN's nodes, etc.), transport latency (downloading the content), and client-side processing (demultiplexing, decoding, rendering).

In a HAS system, the video content is encoded in several representations varying with respect to their media characteristics, such as spatial resolution, frame rate, and compression rate. They are configured by the service provider during the planning phase. Each representation is split into segments, typically containing several seconds of video data such that switching the representation is feasible on each segment boundary. The client issues HTTP requests to download the segments in chronological order, selecting the representation for each of them. After the segment is downloaded, it is stored in the playback buffer until its playback deadline is reached. With live streaming, a segment becomes available for download during the course of the streaming session. If the download is not completed before the playback deadline, the segment is skipped. Since different representations typically have different media bit rates, the client is able to satisfy the latency constraint by dynamically selecting an appropriate representation for each segment. Note that the segment duration affects the client's responsiveness to throughput changes and thus facilitates achieving low latencies. At the same time, however, small segments increase the overhead due to the higher number of HTTP requests and reduce the video compression efficiency due to the decreased GOP size. Typical segment durations lie between 2 and 15 seconds.

Since HTTP offers no means to cancel an ongoing request, the only way to prevent wasting bandwidth by downloading the remaining bytes of a segment whose playback has to be skipped is to shut down the TCP connection. Since opening a new TCP connection is associated with communication overhead, we assume that the client maintains multiple TCP connections, using them in a round-robin manner to keep their internal state, such as congestion window size, and round-trip time (RTT) estimation up-to-date. Whenever a TCP connection is closed, other connections are used, whereas the closed connection is replaced by a new one.

In addition to satisfying latency requirements, one of the main goals of the adaptation is to maximize the QoE. In the following, we define QoE as the quadruple (latency, number of skipped segments, number of quality transitions, average video quality). We use the term *video quality* to refer to the video distortion, which is typically a concave function of the video bit rate [Sullivan and Wiegand 1998]. As stated previously, it is necessary to consider these factors jointly since optimizing any one parameter individually leads to poor QoE. Our approach is to heuristically maximize the average video quality as a function of the triplet: latency, number of skipped segments, and number of quality transitions, which we define as an operating point. Since the duration of the streaming session is not known in advance (the user might quit the session prematurely), the last two values are expressed in relative terms: fraction of skipped segments and fraction of segments that result in a transition. The operating point may be defined by the user, the operating system, the client software, or the content provider. It might depend on various factors, such as the nature of the video, the user context, and the provider's business model.

3.2. Notation

In the following, we introduce the notation used throughout the article. All time-related variables are real valued and represent continuous time. The start time of the recorded

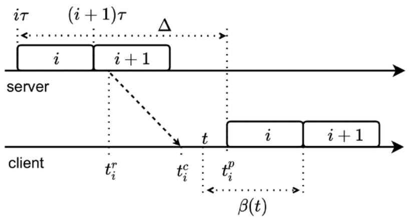

Fig. 1. Illustration of the used notation.

content is $t = 0$. The duration of the video content contained in one segment is assumed to be constant and is denoted by τ. We use the index $i \in \{0, 1, \ldots, n-1\}$ to indicate a particular segment in a stream, whose content covers the time period $[i\tau,\ (i+1)\tau]$, and which becomes available for the download at the time $(i+1)\tau$.

We denote the set of the representations by $\mathcal{R}$, indexed by $j \in \{0, 1, \ldots, |\mathcal{R}|-1\}$. We denote the size in bits and the mean media bit rate (MMBR) of a segment i from the representation j by s_{ij}, respectively $\bar{r}_{ij} = s_{ij}/\tau$. We denote by $\bar{r}_j = \frac{1}{n}\sum_{i=0}^{n-1}\bar{r}_{ij}$ the MMBR's of the representation j. We denote the size and the MMBR's of a downloaded segment i by s_i, respectively $\bar{r}_i = s_i/\tau$, omitting the representation index.

The time when a segment i is requested by the user is denoted by t_i^r. Note that it arises from the maximum of two values: the time when the client finished downloading segment $i-1$ and the time when segment i becomes available at the server. t_i^c denotes the time when the last bit of segment i is received by the user. We denote the target live latency by Δ. Consequently, the playback deadline of segment i is $t_i^p = i\tau + \Delta$. Note that $\Delta - \tau$ constitutes an upper bound on the maximum transport latency, if other latency components are neglected. The maximum transport latency for the individual segments can still be smaller, as it depends on the completion time of the preceding request. We define the playback buffer level at time t as the time until the playback deadline of the next segment whose download is not completed yet: $\beta(t) = \max\{t_i^p \mid t_i^c \leq t\} + \tau - t$. The maximum transport latency for segment i can thus be expressed by $0 < \beta(t_i^r) \leq \Delta - \tau$. Figure 1 provides an illustration.

We denote the fraction of segments skipped until the time t by $\Sigma(t) \in [0, 1]$. When segment i is downloaded and played in representation j different from the representation of the previous successfully downloaded segment, a quality transition occurs. The fraction of segments that were successfully downloaded in a different quality than their predecessors until the time t is denoted by $\Omega(t) \in [0, 1]$.

Note that the traffic generated by a live streaming client might contain interrequest periods during which the client is waiting for the next segment to become available. When computing average application layer throughput, we exclude the interrequest periods to obtain an estimate of the throughput that was actually achieved during a data transmission. We first compute the segment throughput for each downloaded segment i as $\rho_i = s_i/(t_i^c - t_i^r)$. Note that this computation accounts for the round-trip delay. We then compute the average application layer throughput for the time interval $[t_1, t_2]$ as

$$\rho\,(t_1, t_2) = \frac{\sum_{i=l_1}^{l_2} \rho_i \cdot \left| \left[t_i^r, t_i^c\right] \cap [t_1, t_2] \right|}{\sum_{i=l_1}^{l_2} \left| \left[t_i^r, t_i^c\right] \cap [t_1, t_2] \right|}, \tag{1}$$

where l_1 corresponds to the last segment requested before t_1, l_2 is the first segment whose download was completed after t_2, and $|[a, b]| = b - a$. For incomplete downloads, t_i^c must be replaced by the time when the download was interrupted, whereas s_i must be replaced by the number of actually downloaded bytes. For time intervals, for which the denominator equals 0, $\rho(t_1, t_2)$ is not defined.

4. LOLYPOP: ADAPTATION ALGORITHM FOR LOW-DELAY LIVE STREAMING

In this section, we present our design of LOLYPOP, a novel prediction-based adaptation algorithm for low-delay live streaming over HTTP.

4.1. Algorithm Description

As stated in Section 3, the goal of LOLYPOP is to maximize the average video quality as a function of the operating point, defined by the triplet (Δ, Σ, Ω). The input parameters controlling the reached operating point are (i) the target latency Δ, (ii) a control for the fraction of skipped segments $\Sigma^* \in [0, 1]$, and (iii) an upper bound on the (relative) number of quality transitions $\Omega^* \in [0, 1]$. The output of the algorithm is the representation for the next segment to be downloaded. The approach leverages throughput predictions and prediction error estimations to compute the probability P^p_{ij} that the download of segment i in quality j will be completed before its playback deadline. The computation of P^p_{ij} is described in detail in Section 6.5. For now, we assume that P^p_{ij} are given.

Selecting the representation for a segment i is a two-step process. First, LOLYPOP identifies the highest representation j' such that the probability for missing the playback deadline is bounded by Σ^* (i.e., $1 - P^p_{ij'} \leq \Sigma^*$). If no representation satisfies this condition, or if the download success probabilities cannot be computed (e.g., because the session has just started or after an idle period), then j' is set to 0.

In the second step, LOLYPOP computes $\Omega(t)$, the fraction of segments that were played in a different quality than their predecessor. If $\Omega(t) > \Omega^*$ and $j' > j^{\leftarrow}$, where $j^{\leftarrow}$ is the representation of the last successfully downloaded segment $i^{\leftarrow} < i$, representation $j^{\leftarrow}$ is selected to prevent $\Omega(t)$ from further exceeding the upper bound Ω^*. Otherwise, j' is selected. The pseudocode for the described algorithm is presented in Algorithm 1.

ALGORITHM 1: LOLYPOP

```
  Input: $\tau$, $\mathcal{R}$                                   ▷ Video parameters
  Input: $t^r_i$, $t^p_i$, $\Sigma^*$, $\Omega^*$, $T_{\max}$     ▷ Invocation time, playback deadline, config. parameters
  Input: $\Omega(t^r_i)$                                          ▷ Current value for the rel. number of quality transitions
  Input: $(P^p_{ij}, j \in \{0, \dots, |\mathcal{R}| - 1\})$      ▷ Estimated download success probabilities, or –1
  Input: $j^{\leftarrow} \in \{0, \dots, |\mathcal{R}| - 1\}$    ▷ Repr. of the last successf. downloaded segment
  Output: $j^*$                                                   ▷ Selected representation
  Require: $(i+1)\tau \leq t^r_i < t^p_i \leq t^r_i + T_{\max}$  ▷ Segment i available, playback deadline not
                                                                    passed and within prediction horizon
1 if $P^p_{ij} = -1,\ \forall j \in \{0, \dots, |\mathcal{R}| - 1\}$ then   ▷ No estimation available
2 |   $j^* = 0$                                                   ▷ Select lowest representation
3 else                                                            ▷ An estimation of download success probab. is available
4 |   $j' = \max(\{0\} \cup \{ j \in \{0, \dots, |\mathcal{R}| - 1\} \mid 1 - P^p_{ij} \leq \Sigma^*\})$   ▷ Max. repr. satisfying $\Sigma^*$, or 0
5 |   if $\Omega(t^r_i) \leq \Omega^*$ then                        ▷ Transition to a higher representation is allowed
6 |   |   $j^* = j'$                                              ▷ Select the computed representation
7 |   else                                                        ▷ Transition to a higher representation is not allowed
8 |   |   $j^* = \min(j', j^{\leftarrow})$                         ▷ Select the computed representation, if below the previous
```

The intuition for letting LOLYPOP switch to a lower representation even if the upper bound on the quality transitions is exceeded is that blocking a quality decrease can significantly increase the number of skipped segments. According to a large-scale study of user engagement (time before the user quits a streaming session), it is always better to drop the video quality than to let the streaming stall [Conviva 2014].

4.2. Tuning In to the Stream

Starting a new streaming session at time t_0, the client decides which segment i_0 to download first and in which representation. After the download, the clients delays the playback until the playback deadline $t_{i_0}^p = i_0\tau + \Delta$, to fully benefit from the configured target latency Δ. Starting with the newest available segment maximizes the download success probability (there is more time until the playback deadline) but increases the initial delay (the client has to wait longer after the download). In contrast, taking an older segment whose playback deadline is sufficiently far into the future minimizes the initial delay given that the download can be completed in time. LOLYPOP adopts the latter approach, selecting the oldest segment whose playback deadline is at least τ seconds into the future, $i_0 = \min\{i \geq 0 \mid (i+1)\tau \leq t_0 \leq t_i^p - \tau\}$, and downloading it in the lowest quality to minimize the risk of skipping the first segment and thus increasing the initial delay. The expectation is that the available network resources support the download of a segment in its lowest quality in less time than the segment duration, although, due to the small segment duration, the low quality of the first segment should have negligible impact on the QoE of the whole session. If the first segment is downloaded in time, the startup delay $t_{i_0}^p - t_{i_0}$ lies in the interval $[\tau,\ \Delta - \tau]$, which can be seen by using the definition of i_0 and that of the playback deadline $t_{i_0}^p = i_0\tau + \Delta$. LOLYPOP applies the same decision process when the client skips a segment and has to select a segment with which to proceed.

5. TCP THROUGHPUT TRACES

At the core of LOLYPOP is an estimation of the download success probabilities P_{ij}^p. To evaluate the accuracy and error distributions of different prediction methods, but also to evaluate the proposed adaptation algorithm under different network conditions, we required an appropriate dataset. Due to the targeted latency of a few seconds, this dataset had to contain throughput averages computed over relatively small time intervals: 1 second or less. To the best of our knowledge, there existed no such publicly available dataset.

Therefore, we collected a representative set of TCP throughput traces in IEEE 802.11 networks in different environments at various locations throughout Berlin, Germany, and Irbid, Jordan. Our selected locations included highly busy public hotspots, such as airports or conference venues, moderately busy campus hotspots and public hotspots at touristic places, and less busy cells in office and residential environments. The traces were collected using laptops running Ubuntu 13.04 and Ubuntu 14.04 operating systems with default protocol stack configurations, particularly the TCP CUBIC congestion control scheme [Ha et al. 2008]. As senders, we used a server located at the TU Berlin campus (Ubuntu 12.04) and an Amazon EC2 instance located in Ireland (Ubuntu 14.04).

We collected 127 traces of continuous downstream TCP flows lasting between 600 and 3,600 seconds each. To focus on the challenging scenarios, we removed traces with a Coefficient of Variation (CV) below 0.1, leaving 92 traces with a total length of 45 hours. More information about the traces, an example figure depicting the throughput of one complete trace, and the download link are provided in the online appendix.

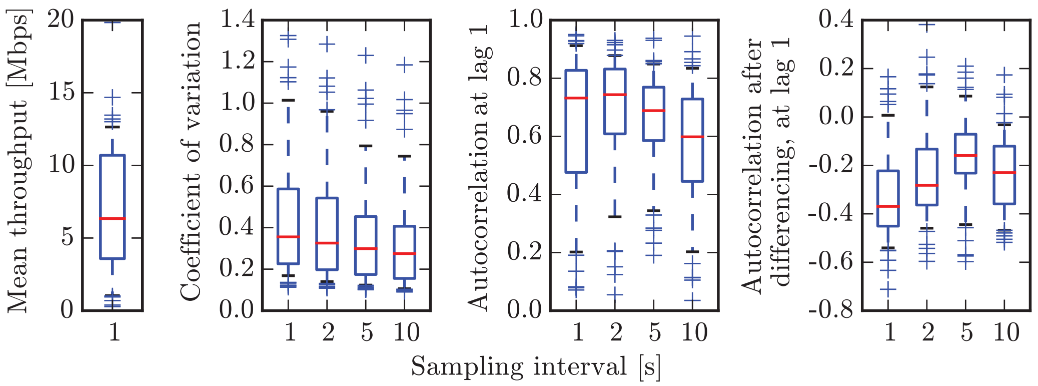

Fig. 2. Trace statistics: mean throughput (equal for all sampling intervals), CV, autocorrelation at lag 1, and autocorrelation, after differencing, at lag 1. Horizontal line, median; box, quartiles; whiskers, 0.5 and 0.95 quantiles; flier points, outliers.

Figure 2 provides an illustration of the throughput variability and temporal correlation that are among the main factors affecting the predictability. The figure shows box plots for the mean throughput, CV (standard deviation divided by the mean), autocorrelation at lag 1, and autocorrelation after differencing at lag 1. Autocorrelation at lag 1 shows how probable it is that a large throughput value is followed by another large value (autocorrelation close to 1) or by a small value (autocorrelation close to –1). A value close to 0 indicates a lack of temporal correlation. Autocorrelation after differencing quantifies the correlation of throughput changes. The computed statistics confirm that our traces cover a broad range of network conditions. A mean throughput between 4 and 11 Mbps is noted for 50% of traces, whereas the mean lies between 1 and 13 Mbps for 90% of traces. The range of throughput fluctuations, represented by the CV, varies approximately from 0.1 to 1.3. An interesting observation is that 75% of traces show autocorrelation values of over 0.6 at a sampling interval of 2 seconds, whereas 95% still have autocorrelation values over 0.3, indicating significant temporal correlation between subsequent measurements. At the same time, the time series of throughput changes exhibits a strong negative autocorrelation, indicating that a throughput increase is likely to be followed by a throughput decrease.

6. SHORT-TERM TCP THROUGHPUT PREDICTION

In this section, we present our approach to estimating the download success probabilities required by LOLYPOP. It is based on predicting TCP throughput and estimating the relative prediction error distribution.

6.1. Methodology

Our goal is to estimate the probabilities P_{ij}^p that the download of s_{ij} bytes, requested at time t_i^r, will be completed by the time t_i^p. We achieve this by using a time series prediction complemented by estimating the relative prediction error distribution. A time series prediction method uses several past values to compute one or several future values. Thus, from $\rho(t-(k+1)T, t-kT)$, $k \in \{0,\ldots,k_p-1\}$, it computes predictions $\hat{\rho}(t+kT, t+(k+1)T)$, $k \in \{0,\ldots,k_f-1\}$, where T is the averaging interval.

As described in Section 3, the upper bound on the download duration $t_i^p - t_i^r$ for segment i takes values from the range $(0, \Delta - \tau]$, depending on the completion time of the preceding download. We therefore have the following two options to compute predictions. We can fix T and, whenever $t_i^p - t_i^r > T$, compute a prediction for multiple steps into the future, or we compute predictions using multiple values for T and then use the smallest one such that $T \geq t_i^p - t_i^r$. In the course of the study, we observed that the latter approach performs significantly better. Consequently, we focus on predictions on multiple time scales, for one step into the future. We focus on time scales from 1 to 10 seconds because of their relevance to low-delay streaming.

Given a prediction $\hat{\rho}(t_1,\, t_2)$ and the corresponding measured throughput $\rho(t_1,\, t_2)$, we compute the relative prediction error as

$$\epsilon\,(t_1,\, t_2) = \frac{|\max\left(\hat{\rho}(t_1,\, t_2),\ \rho_{\min}\right) - \max\left(\rho(t_1,\, t_2),\ \rho_{\min}\right)|}{\max\left(\rho(t_1,\, t_2),\ \rho_{\min}\right)}. \tag{2}$$

Here, the maximum operator prevents a distortion of results whenever $\rho \approx 0$ or $\hat{\rho} \leq 0$. In the following, we set $\rho_{\min} = 10$ kbps. We separately evaluate the overestimation and the underestimation errors, due to their different error ranges ($(0, \infty)$ and $(0, 1]$, respectively) and due to their different impacts on the adaptation. An overestimation increases the risk of skipping a segment, which has the strongest impact on QoE. An underestimation decreases the risk of interruptions but reduces the video quality.

6.2. Prediction Methods

We evaluated a number of time series prediction methods, including simple moving average (SMA), linear extrapolation, cubic smoothing splines (CSS), several flavors of exponential and double exponential smoothing, autoregressive integrative moving average (ARIMA), and several machine learning–based methods [Chatfield 2003]. In the following, we present results for three methods: SMA, linear extrapolation, and double exponential smoothing (Holt-Winters). A description of these methods is provided in the online appendix to this article. We use the abbreviation ⟨type⟩:⟨k_p⟩:⟨parameters⟩, where ⟨type⟩ is the name of the method, k_p is the number of past throughput values used as input, and ⟨parameters⟩ include further optional configuration parameters. For example, SMA:k_p:ar denotes SMA with arithmetic mean, and HW:k_p:mse denotes Holt-Winters with mean squared error (MSE) for parameter tuning.

6.3. Evaluation of the Prediction Accuracy

We evaluate the prediction methods in two steps. First, we compare the relative prediction errors over the joint data set from all traces to identify the method that performs best over a broad range of network environments. In the second step, we evaluate the prediction accuracy variation among the individual traces.

To compare the prediction accuracy over the complete dataset, we computed the 0.2, 0.5, and 0.9 quantiles of the error. For example, a 0.2 quantile of 0.3 means that in 20% of all data points the error is below 0.3. Another example: a 0.9 quantile of 0.8 means that less than 10% of data points have an error above 0.8. The results for the sampling interval of 5 seconds are shown in Figure 3. We observe that SMA:1:ar has the best performance, except for the 0.9 quantile of the underestimation error, where SMA:10:ar is the best-performing method. For 50% of the data points, SMA:1:ar results in an overestimation error that does not exceed 10%, whereas only less than 10% of data points have an error of 80% and larger. This is somewhat surprising, as SMA:1:ar is the most naive method that uses the most recent measurement as prediction. It also has a much lower computational complexity than methods such as Holt-Winters due to the optimizations involved in tuning the configuration parameters of the latter for every new prediction. It seems that taking into account the trend in the past measurements does not improve the prediction quality. This is consistent with the observation that in many traces the differences in subsequent throughput measurements show a negative correlation, as depicted in Figure 2. In addition, methods using a small number of history points implicitly detect level shifts and do not propagate outliers. These two

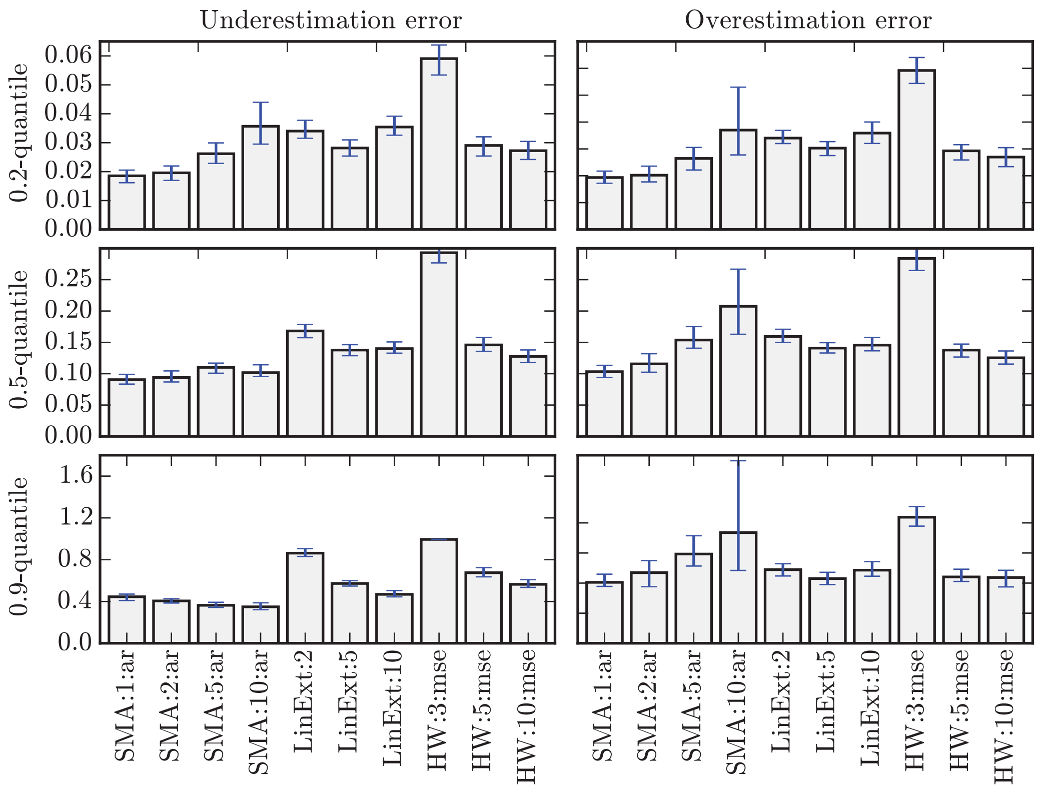

Fig. 3. Relative prediction error quantiles for the complete dataset, on the time scale of 5 seconds. The error bars show confidence intervals for the confidence level of 0.95 (a varying subsampling factor has been used for decorrelation [Le Boudec 2015]).

issues were reported to be known challenges in TCP throughput prediction [He et al. 2007].

In the second step, we evaluate how the prediction accuracy varies over the individual traces. For each trace and method, we compute the fractions of predictions with a relative error less than 0.2, 0.5, and 1.0. The Empirical Cumulative Distribution Function (ECDF) of these fractions over the individual traces for the sampling interval of 5 seconds is shown in Figure 4. The first, second, and third columns shows for each trace the fraction of measurements with a relative error below 0.2, 0.3, and 0.5, respectively. For example, the point (0.8, 0.3) on the solid line in the middle column, bottom row, represents a trace, where 80% of the overestimations have a relative error of 0.5 or less. Points below (y-value less than 0.3) correspond to traces that have a worse performance, whereas points above (y-value greater than 0.3) correspond to traces with a better performance. The fact that the graph passes through point (0.8, 0.3) means that in 70% of traces (sampled at 5 seconds), less than 20% of the overestimations have a relative error of 0.5 or more.

From Figure 4, we observe that the accuracy strongly varies across traces. There are traces where 90% of overestimations have an error less than 20%, whereas 100% have an error less than 50%. A video client might account for an error of this magnitude by using a fixed safety margin—that is, by always selecting a media bit rate that is 20% smaller than the predicted throughput. There are, however, traces where almost 60% of the overestimations have a relative error of greater than 50%, whereas more

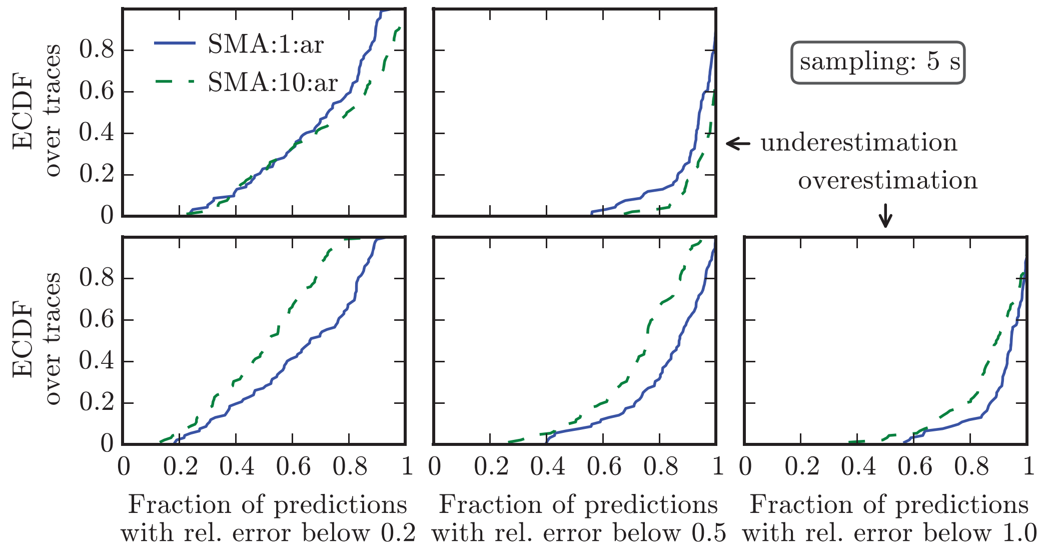

Fig. 4. Performance of SMA:1:ar and SMA:10:ar for individual traces.

than 40% still have an error greater than 100%. Setting a high fixed safety margin to account for such "bad" traces would result in a significant underutilization of the network resources, a lower media bit rate, and thus a lower QoE, in the "well-behaving" traces. On the other hand, selecting a low fixed safety margin would increase the total number of skipped segments in the bad traces. Consequently, we have to complement a time series prediction with an estimation of the prediction error distribution.

Finally, in all of the studied traces, we observed that the probability for occurrences of underestimations and overestimations is well balanced on all time scales. Both occur in approximately 50% $\pm$ 5% of cases. At the same time, they exhibit a significant temporal correlation: the probability that an underestimation is followed by an overestimation and vice versa is significantly greater than 50% for most traces for all time scales, exceeding 80% or even 90% in some cases. This observation is directly related to the negative correlation of the throughput after differencing, depicted in Figure 2. The distribution of per-trace values is depicted in the online appendix.

6.4. Estimating the Relative Prediction Error

As we have seen, it is not sufficient to perform a point prediction of the throughput because of its potentially high uncertainty and varying accuracy across network environments. There are approaches that allow to explicitly predict a distribution, such as the Gaussian process method [Rasmussen and Williams 2006]. However, approaches such as SMA do not have this capability. Therefore, we complement the predicted value by an estimate of the relative prediction error distribution. A straightforward approach is to use the ECDF of past prediction errors and account for the long-term nonstationarities by discarding sufficiently old values. Another approach is to select a distribution type and fit its parameters dynamically from the data. In our evaluation, we will use the former method, as it results in good performance and the latter method is much more resource consuming. In the online appendix, we present our results on fitting several well-known distribution types to the relative prediction errors: exponential, normal, logistic, and Lomax (shifted Pareto) [Johnson et al. 1994]. We observe that the Lomax distribution provides the best fit. We therefore recommend using the Lomax distribution to model prediction errors when evaluating adaptation approaches with synthetic data as done, for example, by Yin et al. [2014].

6.5. Estimating Download Success Probabilities

In this section, we describe our approach to estimating the probabilities ($P^p_{ij}, j \in \{0, \ldots, |\mathcal{R}| - 1\}$) that at time t^r_i s_{ij} bits can be downloaded before the playback deadline t^p_i. We denote by $T_{\max} \in \mathbb{N}$ the maximum prediction horizon in seconds. At time t, the client computes the average application layer throughput (1) for the time intervals $[t - T, t]$ for $T \in \{1, \ldots, T_{\max}\}$, followed by computing the predictions for the time intervals $[t, t + T]$. If the throughput cannot be computed, no prediction is provided. Finally, the client computes the relative prediction error for the intervals $[t - T, t]$ as

$$\tilde{\epsilon}(t - T, t) = \frac{\max(\hat{\rho}(t - T, t), \rho_{\min}) - \max(\rho(t - T, t), \rho_{\min})}{\max(\rho(t - T, t), \rho_{\min})}. \tag{3}$$

In contrast to (2), $\tilde{\epsilon}(t - T, t) \in (-1, \infty)$ is defined without taking the absolute value.

We assume that the predictions are computed every second, so at time t^r_i, the most recent predictions are dated at time $\lfloor t^r_i \rfloor$. To calculate the download success probabilities, the client determines the smallest time interval containing $[t^r_i, t^p_i]$ for which a prediction is available. Note that it is not necessarily $[\lfloor t^r_i \rfloor, \lceil t^p_i \rceil]$, because due to the distribution of interrequest delays or due to a link outage, a prediction for this time interval might not be available. Let $t^\pi_i, T^* \in \mathbb{N}$ be determined such that $[t^\pi_i, t^\pi_i + T^*]$ is the shortest time interval containing $[t^r_i, t^p_i]$ for which a prediction is available, and let $\hat{\rho}_i = \hat{\rho}(t^\pi_i, t^\pi_i + T)$ be the corresponding throughput prediction. $\epsilon_i = \epsilon(t^\pi_i, t^\pi_i + T)$ and $\tilde{\epsilon}_i = \tilde{\epsilon}(t^\pi_i, t^\pi_i + T)$ shall denote the relative prediction errors, as defined in (2) and (3). Further, we denote by $\Phi^u_i(\epsilon_i)$ and $\Phi^o_i(\epsilon_i)$ the estimated cumulative distribution function (CDF) of the underestimation and overestimation errors for $\hat{\rho}_i$, computed at t^π_i. Finally, $P^u_i \in [0, 1]$ shall denote the relative frequency of underestimations. With the introduced notation, the ECDF's for $\tilde{\epsilon}_i$ is given by

$$\Phi_i(\tilde{\epsilon}_i) = \begin{cases} P^u_i \cdot \Phi^u_i(\epsilon_i) & \text{for } \tilde{\epsilon}_i < 0 \\ P^u_i + \left(1 - P^u_i\right) \cdot \Phi^o_i(\epsilon_i) & \text{otherwise}. \end{cases} \tag{4}$$

Consequently, the download success probability P^p_{ij} can be estimated as

$$P^p_{ij} = P\left[\frac{s_{ij}}{t^p_i - t^r_i} \le \frac{\hat{\rho}_i}{1 + \tilde{\epsilon}_i}\right] = \Phi_i\left(\frac{\hat{\rho}_i\left(t^p_i - t^r_i\right)}{s_{ij}} - 1\right). \tag{5}$$

7. EVALUATION

We evaluated the performance of LOLYPOP using our collected throughput traces and compared it against the state-of-the-art algorithm FESTIVE [Jiang et al. 2014] as a baseline. The setting and results are presented in the following.

7.1. Evaluation Setting

We implemented both LOLYPOP and FESTIVE in a streaming client prototype, equipped with a feature that allowed it to be executed in virtual time using a throughput trace file as input, thus allowing for a simulative evaluation using collected traces.

We used Big Buck Bunny[3] as video content, which is an animated movie of 10 minutes duration. This video was selected due to the availability of raw video data, allowing us to generate representations with high MMBRs. We used the H.264/MPEG-4 AVC

[3]http://www.bigbuckbunny.org.

[Wiegand et al. 2003] compression format to encode nine representations with MMBRs distributed between 100 and 20,000 kbps with exponentially increasing intervals: 101, 194, 377, 730, 1,415, 2,743, 5,319, 10,314, and 20,000 kbps. The encoding was performed using the avconv[4] utility using two passes, with a configuration targeting at low to moderate MMBR's variations across the individual segments.

In general, the video distortion represented by, for example, the peak signal-to-noise ratio (PSNR) is a logarithmic-shaped function of the bit rate [Sullivan and Wiegand 1998]. The chosen exponentially spaced MMBRs therefore result in the video quality being roughly proportional to the representation index. Consequently, in our evaluation, we use the representation index as a measure of the video quality.

The evaluation was performed using an upper bound on the transport latency of 3 seconds, corresponding to 1.5 times the segment duration. Neglecting segmentation overhead at the server and decoding overhead at the client, this corresponds to an overall live latency of $\Delta = 5$ seconds. Each streaming session lasted for 5 minutes.

We evaluated LOLYPOP with different values for the configuration parameters Σ^* and Ω^*. The goal was to explore the range of operating points with $\Sigma \in [0, 0.1]$ and $\Omega \in [0, 0.5]$. We used $\Sigma^* \in \{0.005, 0.01, 0.02, 0.03, 0.04, 0.05, 0.06, 0.07, 0.1, 0.15, 0.2, 0.25, 0.3, 0.35, 0.4, 0.45, 0.5, 0.55, 0.6, 0.65, 0.7, 0.75, 0.8, 0.85, 0.9, 0.95\}$ and $\Omega^* \in \{0.001, 0.005, 0.008, 0.01, 0.02, 0.03, 0.04, 0.05, 0.06, 0.07, 0.08, 0.09, 0.1, 0.15, 0.2, 0.3, 0.5\}$. In total, we evaluated 442 configurations. Note that we used Σ^* values that are much higher than the values for Σ that we want to achieve. This is due to the observation that tight restrictions on the number of quality transitions Ω^* results in much lower numbers of skipped segments than the value used for Σ^*.

We evaluated FESTIVE with a broad range of values around the default configuration evaluated by Jiang et al. [2014]. We vary the values for α (controlling the trade-off between the average quality and the quality fluctuations), p (safety margin between the estimated bandwidth and the selected MMBR's), and k (controlling the amount of quality fluctuations by enforcing a minimum distance between quality transitions). We used $\alpha \in \{5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 16, 17, 18, 19, 20\}$, $p \in \{0.4, 0.45, 0.5, 0.55, 0.6, 0.65, 0.7, 0.75, 0.8, 0.85, 0.9, 0.95\}$, and $k \in \{1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 15, 20, 30, 40, 50\}$. In total, we evaluated 2,880 configurations. We disabled the randomizer feature of FESTIVE because it requires delaying requests. With low-delay streaming, the randomizer feature can lead to an increased number of skipped segments. We did not relax the restriction of FESTIVE that it switches the representation at most one step at a time, as we considered it one of its core features. Finally, we would like to point out that FESTIVE bases its decisions on the knowledge of the MMBR's of a representation, whereas LOLYPOP uses the segment size of the next segment.

7.2. Evaluation Results

The evaluation goals are to understand the dependency of the reached operating point on the algorithm configuration, explore the region of reachable operating points, and evaluate the average video quality as a function of the operating point.

First, we study the dependency of the reached operating point, specifically of (Σ, Ω), on the input parameters Σ^* and Ω^*. Figure 5 (left) illustrates the ability of LOLYPOP to satisfy the upper bound on the number of quality transitions Ω^* by depicting Ω as a function of Ω^* exemplarily for $\Sigma^* = 0.1$. The graphs for other values of Σ^* are almost identical and omitted. We observe that LOLYPOP is able to enforce the upper bound on the number of quality transitions quite accurately. One reason for the slight overshoot is that we always allow downward quality transitions. In addition, note that a value of $\Omega = 0.01$ means that during the whole streaming session, there are only three quality

[4]http://libav.org/avconv.html.

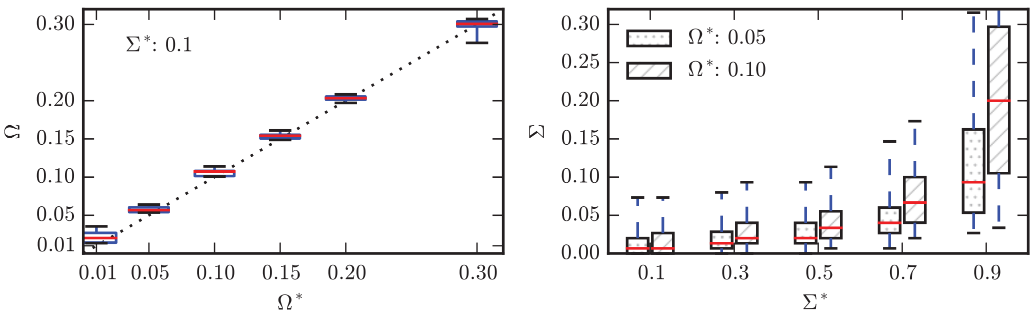

Fig. 5. Ω as function of Ω* (left), and Σ as function of Σ* (right), for LOLYPOP. Horizontal line, median; box, quartiles; whiskers, 0.05 and 0.95 quantiles.

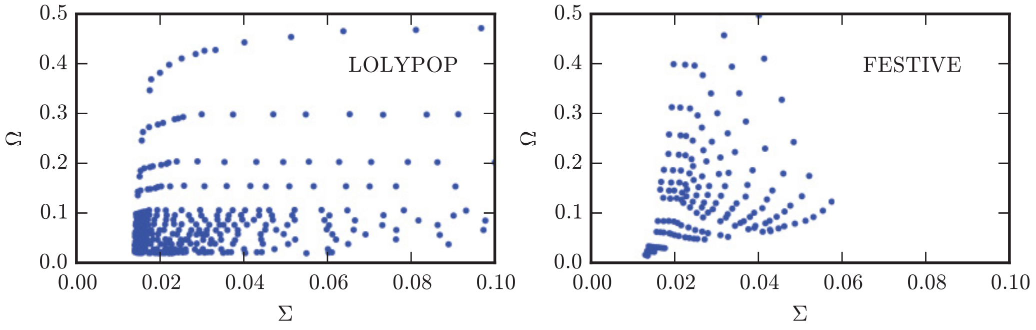

Fig. 6. Scatter plots of covered (Σ, Ω) regions for LOLYPOP (left) and FESTIVE (right).

transition, which in a wireless network is an extremely low value. Figure 5 (right) illustrates the dependency of Σ on Σ* for two values of Ω*: 0.05 and 0.1. We observe that Σ is significantly lower than Σ* and that a lower value for Ω* decreases Σ even further. The intuition behind that is that whenever Ω* is exceeded during the course of a streaming session, only downward quality transitions are permitted.

Next, we evaluate the region of reachable operating points. The broader is this region, the more flexible the algorithm can be tuned to the QoE requirements defined for a streaming session. Figure 6 shows the reached (Σ, Ω) pairs for LOLYPOP (left) and FESTIVE (right). We observe that the studied LOLYPOP configurations cover a broader range of (Σ, Ω) values, enabling a more flexible adjustment to user and/or application profiles. Note that a low value of Σ and/or Ω alone is not an indicator of high QoE, as it might be achieved by selecting an unnecessarily low video quality.

The fact that Σ values below 0.01 were not achieved by either algorithms is in part explained by the throughput outages of several seconds duration contained in some traces. To quantify these "unavoidable" fractions of skipped segments, we simulate streaming sessions using an adaptation algorithm that always selects the lowest quality. We observed that out of 92 used traces, 66 support streaming at lowest quality without skipped segments. Furthermore, 7 traces had a Σ below 0.01, further 10 below 0.05, further 7 below 0.1, one had 0.12, and one had the highest Σ of 0.15.

As a main result of the evaluation, we express the average video quality as a function of the operating point. Figure 7 shows the mean representation (index) as a function of the number of skipped segments for different numbers of quality transitions. For different values of Σ, we first determine the configuration that (i) achieves the highest average video quality over all traces, (ii) whose average fraction of skipped segments

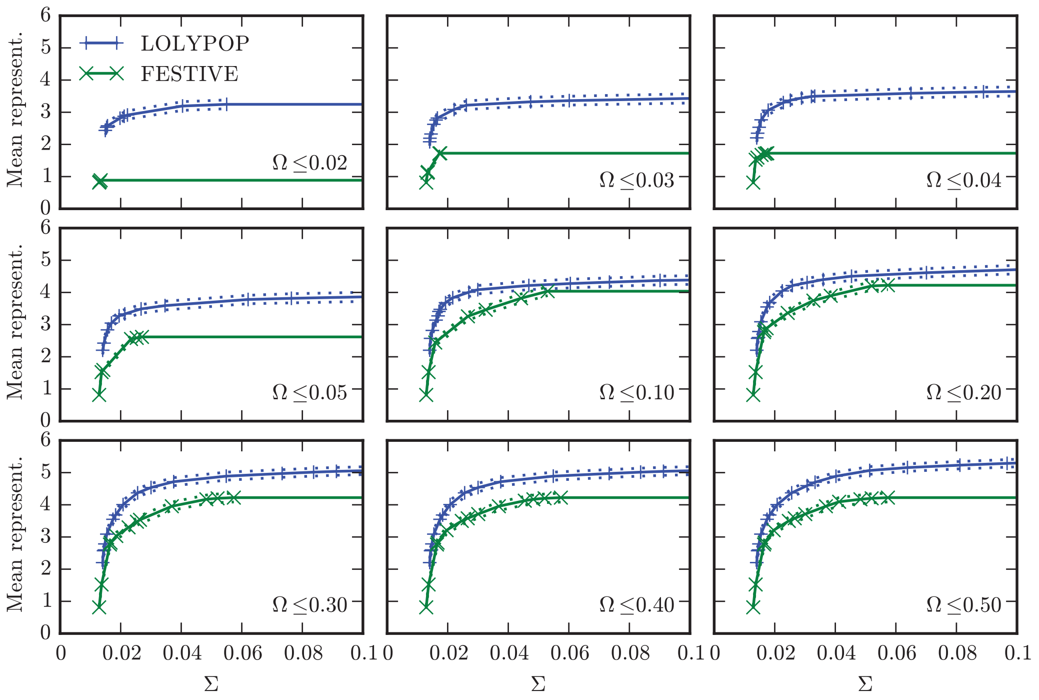

Fig. 7. Average video quality as a function of the number of skipped segments Σ for different numbers of quality transitions Ω. Dashed lines represent the confidence interval for the confidence level 0.95.

is less than or equal to Σ, and (iii) whose average number of quality transitions is less than or equal to Ω, where $\Omega \in \{0.02, 0.03, 0.04, 0.05, 0.1, 0.2, 0.3, 0.4, 0.5\}$. The convex hulls of the resulting curves are depicted in Figure 7. It is accompanied by the confidence interval for the confidence level of 0.95, computed after subsampling the data with a subsampling factor of 0.1 to remove temporal correlation [Le Boudec 2015].

We observe that LOLYPOP achieves a higher average quality at all operating points. The difference is particularly pronounced for small numbers of quality transitions, where LOLYPOP achieves an up to three times higher average quality. An interesting observation is that all plots have a more or less pronounced "knee," after which the curve goes into saturation and the quality does not increase significantly. In contrast, before the knee, a small increase in the number of skipped segments can bring a huge increase in video quality. In the evaluated network environments, the knee is typically slightly below $\Sigma = 0.02$. In other words, accepting 0.5% to 1% more skipped segments, which corresponds to one to two more skipped segments every 400 seconds, can result in an up to twofold improvement in video quality (e.g., for $\Omega \leq 0.2$).

Figure 7 shows mean values over all 92 traces; however, we generated similar plots for the individual traces. In 31 traces, all 9 considered Ω thresholds resulted in both curves having the same ranges and could thus be compared pointwise. In 21 out of the 31 traces, all 9 curves for LOLYPOP were pointwise strictly greater than the corresponding curves for FESTIVE, whereas there existed no traces where all 9 curves were pointwise greater for FESTIVE. Furthermore, to perform a trace-by-trace comparison across all traces, including those in which some curves had different ranges or were intersecting, we compared the integrals of the curves. This comparison revealed that (e.g., for $\Omega = 0.04$) in 82% of traces, LOLYPOP had a higher integral than FESTIVE; in 16% of traces,

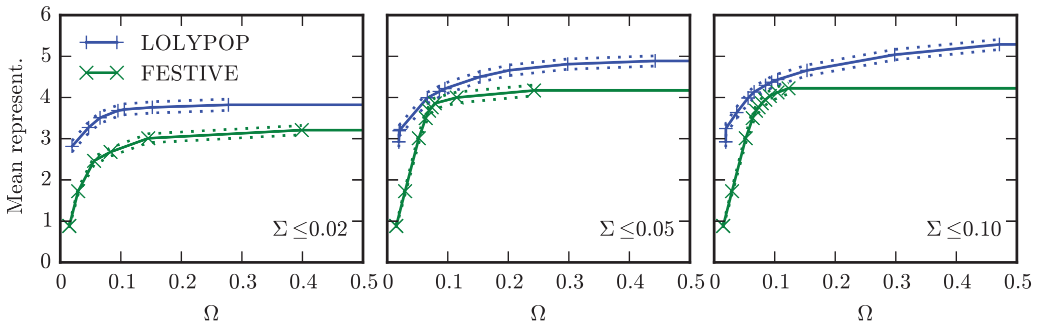

Fig. 8. Average video quality as a function of the number of quality transitions Ω for different numbers of skipped segments Σ. Dashed lines represent the confidence interval for the confidence level 0.95.

FESTIVE had a higher integral; and in the remaining traces, the values were equal. The corresponding values for $\Omega \in \{0.02, 0.03, 0.04, 0.05, 0.1, 0.2, 0.3, 0.4, 0.5\}$ are (53%, 38%), (76%, 22%), (82%, 16%), (76%, 22%), (78%, 20%), (86%, 12%), (87%, 11%), (87%, 11%), (89%, 9%). We thus observe that for all considered Ω thresholds, in the majority of traces, the performance of LOLYPOP averaged over the considered range of Σ values is higher than the performance of FESTIVE.

Similarly to Figure 7, Figure 8 presents plots of quality versus quality transitions for different levels of skipped segments. Here, we observe a similar situation, albeit the knee effect is less pronounced in the case of LOLYPOP due to the relatively high achieved video quality for low values of Ω.

Finally, Figure 9 depicts four example runs illustrating the behavior of LOLYPOP with different configurations. For each run, three plots are shown. The top one depicts the throughput and segment MMBR's. The middle one depicts the adaptation trajectory and the mean representation. The bottom one depicts the buffer level at playback deadlines (a value of 0 results in a skipped segment). In the three upper left subplots, we see a run with low values for both Σ^* and Ω^*. Setting Ω^* to 0.001, we effectively restrict the number of upward transitions to 1, as the streaming session has fewer than 1,000 segments. We observe that LOLYPOP reacts not only to decreased throughput, as seen between 30 and 50 seconds, but also to increased uncertainty in throughput dynamics, as seen after the strong downward fluctuation at 170 seconds. A higher Σ^*, as seen in the top right subplots, results in a more aggressive behavior by accepting higher probabilities for skipping a segment, achieving a higher average quality. The two sets of subplots at the bottom depict runs with $\Omega^* = 0.1$. Here, the client is allowed to have more quality transitions, resulting in further improvement of the average video quality.

8. CONCLUSION

In the presented study, we addressed the problem of maximizing QoE for HTTP-based adaptive low-delay video streaming. We proposed LOLYPOP, an adaptation algorithm designed to operate with a transport latency of a few seconds over wireless access links. LOLYPOP leverages TCP throughput predictions on time scales from 1 to 10 seconds. We studied the performance of several time series prediction methods using traces from various network environments. We observed that the most naive approach, using the last sample as prediction for the future, has the highest accuracy on all considered time scales. We also observed that the quantiles of the prediction error distribution strongly vary among considered traces requiring a dynamic estimation of the error distribution

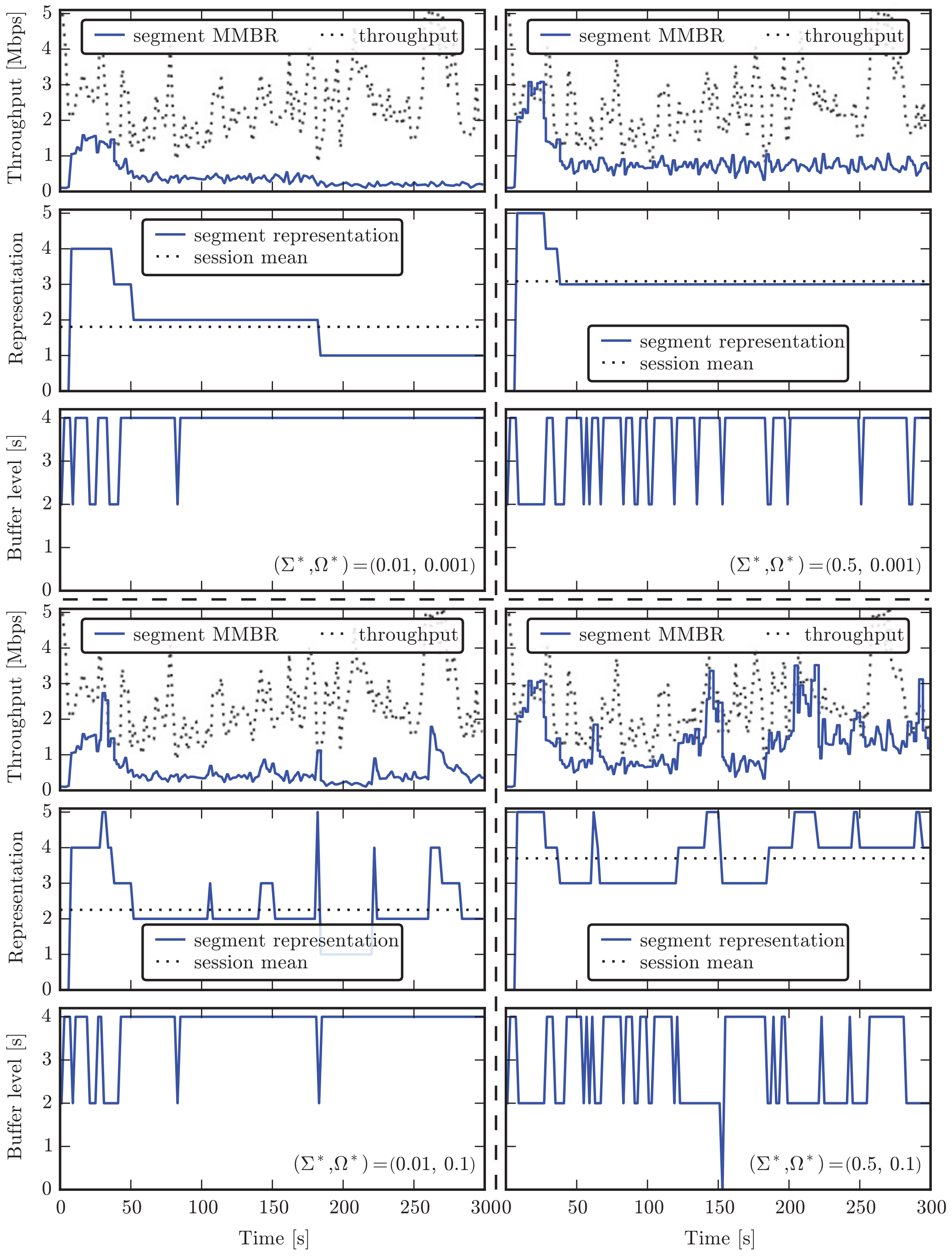

Fig. 9. Four example runs with different algorithm configurations.

for each streaming session individually. We integrated LOLYPOP and a state-of-the-art adaptation algorithm, FESTIVE, with a streaming client prototype and evaluated them using the collected throughput traces. We observed that LOLYPOP reaches a broader range of operating points and outperforms the baseline approach with respect to the mean video quality at each of these operating points by up to a factor of 3.

Our ongoing and future work includes studying the benefits of using cross-layer and context information (i) to adjust the configuration parameters, particularly Σ^*, to achieve a target level of skipped segments, and (ii) to further improve prediction accuracy. Furthermore, we plan to evaluate how Δ can be dynamically tuned to achieve minimum latency without dropping the QoE. Additionally, we plan to evaluate the potential to further reduce the latency by reducing the video segment duration.

ELECTRONIC APPENDIX

The electronic appendix for this article can be accessed in the ACM Digital Library.

REFERENCES

Abdallah S. Abdallah and Allen B. Mackenzie. 2015. A cross-layer controller for adaptive video streaming over IEEE 802.11 networks. In *Proceedings of IEEE International Conference on Communications (ICC'15)*. 6797–6802.

Cristina Aurrecoechea, Andrew T. Campbell, and Linda Hauw. 1998. A survey of QoS architectures. *Multimedia Systems* 6, 3, 138–151.

Athula Balachandran, Vyas Sekar, Aditya Akella, Srinivasan Seshan, Ion Stoica, and Hui Zhang. 2012. A quest for an Internet video quality-of-experience metric. In *Proceedings of ACM Workshop on Hot Topics in Networks (HotNets'12)*. 97–102.

Athula Balachandran, Vyas Sekar, Aditya Akella, Srinivasan Seshan, Ion Stoica, and Hui Zhang. 2013. Developing a predictive model of quality of experience for Internet video. In *Proceedings of the ACM SIGCOMM Conference (SIGCOMM'13)*. 339–350.

Dilip Bethanabhotla, Giuseppe Caire, and Michael J. Neely. 2015. Adaptive video streaming for wireless networks with multiple users and helpers. *IEEE Transactions on Communications* 63, 1, 268–285.

Ayub Bokani, Mahbub Hassan, and Salil Kanhere. 2013. HTTP-based adaptive streaming for mobile clients using Markov decision process. In *Proceedings of the International Packet Video Workshop (PV'13)*.

Jorge Carapinha, Roland Bless, Christoph Werle, Konstantin Miller, Virgil Dobrota, Andrei Bogdan Rus, Heidrun Grob-Lipski, and Horst Roessler. 2010. Quality of service in the future Internet. In *Proceedings of the ITU-T Kaleidoscope Conference*.

Sharon Carmel, Tzur Daboosh, Eli Reifman, Naftali Shani, Ziv Eliraz, Dror Ginsberg, Edan Ayal, and Kfar Saba. 2002. Network Media Streaming. Patent No. US 6389473.

Christopher Chatfield. 2003. *The Analysis of Time Series: An Introduction*. Taylor & Francis, Abingdon, UK.

Zhigang Chen, See-Mong Tan, Roy H. Campbell, and Yongcheng Li. 1995. Real time video and audio in the World Wide Web. In *Proceedings of the International World Wide Web Conference*.

Cisco. 2014. *Cisco Visual Networking Index: Forecast and Methodology, 2013–2018*. White Paper. Cisco Systems, San Jose, CA.

Maxim Claeys, Steven Latre, Jeroen Famaey, and Filip De Turck. 2014. Design and evaluation of a self-learning HTTP adaptive video streaming client. *IEEE Communications Letters* 18, 4, 716–719.

ComScore. 2014. *U.S. Digital Future in Focus*. White Paper. ComScore, Reston, VA.

Conviva. 2014. *Viewer Experience Report*. White Paper. Conviva, Foster City, CA.

Conviva. 2015. *Internet TV: Bringing Control to Chaos*. White Paper. Conviva, Foster City, CA.

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

Sangtae Ha, Injong Rhee, and Lisong Xu. 2008. CUBIC: A new TCP-friendly high-speed TCP variant. *ACM SIGOPS Operating Systems Review* 42, 5, 64–74.

Jia Hao, Roger Zimmermann, and Haiyang Ma. 2014. GTube: Geo-predictive video streaming over HTTP in mobile environments. In *Proceedings of the ACM Multimedia Systems Conference (MMSys'14)*. 259–270.

Qi He, Constantinos Dovrolis, and Mostafa Ammar. 2007. On the predictability of large transfer TCP throughput. *Computer Networks* 51, 14, 3959–3977.

Myles Hollander, Douglas A. Wolfe, and Eric Chicken. 2014. *Nonparametric Statistical Methods* (3rd ed.). Wiley, New York, NY.

T. Hossfeld, S. Egger, R. Schatz, M. Fiedler, K. Masuch, and C. Lorentzen. 2012. Initial delay vs. interruptions: Between the devil and the deep blue sea. In *Proceedings of the Workshop on Quality of Multimedia Experience (QoMEX'12)*.

ITU-T. 2008a. *Definition of Terms Related to Quality of Service (ITU-T E.800)*. Recommendation. ITU-T, Geneva, Switzerland.

ITU-T. 2008b. *Vocabulary for Performance and Quality of Service, Amendment 2: New Definitions for Inclusion in Recommendation ITU-T P.10/G.100*. Recommendation. ITU-T, Geneva, Switzerland.

ITU-T. 2014. *Requirements for Low-Latency Interactive Multimedia Streaming (ITU-T F.746.1)*. Recommendation. ITU-T, Geneva, Switzerland.

Dmitri Jarnikov and Tanr Özçelebi. 2011. Client intelligence for adaptive streaming solutions. *Signal Processing: Image Communication* 26, 7, 378–389.

Junchen Jiang, Vyas Sekar, and Hui Zhang. 2014. Improving fairness, efficiency, and stability in HTTP-based adaptive video streaming with FESTIVE. *IEEE/ACM Transactions on Networking* 22, 1, 326–340.

Norman Lloyd Johnson, Samuel Kotz, and Narayanaswamy Balakrishnan. 1994. *Continuous Univariate Distributions* (2nd ed.). Wiley, New York, NY.

Hemant Kanakia, Partho P. Mishra, and Amy R. Reibman. 1995. An adaptive congestion control scheme for real time packet video transport. *IEEE/ACM Transactions on Networking* 3, 6, 671–682.

Hieu Le, Arash Behboodi, and Adam Wolisz. 2015a. Quality driven resource allocation for adaptive video streaming in OFDMA uplink. In *Proceedings of the IEEE 26th Annual International Symposium on Personal, Indoor, and Mobile Radio Communications (PIMRC'15)*. 1277–1282.

Hung T. Le, Duc V. Nguyen, Nam Pham Ngoc, Anh T. Pham, and Truong Cong Thang. 2013. Buffer-based bitrate adaptation for adaptive HTTP streaming. In *Proceedings of the International Conference on Advanced Technologies for Communications (ATC'13)*. 33–38.

Hung T. Le, Hai N. Nguyen, Nam Pham Ngoc, Anh T. Pham, Hoa Le Minh, and Truong Cong Thang. 2015b. Quality-driven bitrate adaptation method for HTTP live-streaming. In *Proceedings of the IEEE International Conference on Communication Workshop (ICCW'15)*. 1771–1776.

Jean-Yves Le Boudec. 2015. *Performance Evaluation of Computer and Communication Systems* (Version 2.3). EPFL Press, Lausanne, Switzerland.

Blazej Lewcio, Benjamin Belmudez, Theresa Enghardt, and Sebastian Möller. 2011. On the way to high-quality video calls in future mobile networks. In *Proceedings of the International Workshop on Quality of Multimedia Experience (QoMEX'11)*. 43–48.

Baochun Li, Zhi Wang, Jiangchuan Liu, and Wenwu Zhu. 2013. Two decades of Internet video streaming: A retrospective view. *ACM Transactions on Multimedia Computing, Communications, and Applications* 9, 1, 1–20.

Zhi Li, Ali C. Begen, Joshua Gahm, Yufeng Shan, Bruce Osler, and David Oran. 2014a. Streaming video over HTTP with consistent quality. In *Proceedings of the 5th ACM Multimedia Systems Conference (MMSys'14)*. 248–258.

Zhi Li, Xiaoqing Zhu, Josh Gahm, Rong Pan, Hao Hu, Ali C. Begen, and Dave Oran. 2014b. Probe and adapt: Rate adaptation for HTTP video streaming at scale. *IEEE Journal on Selected Areas in Communications* 32, 4, 719–733.

Chenghao Liu, Imed Bouazizi, and Moncef Gabbouj. 2011. Rate adaptation for adaptive HTTP streaming. In *Proceedings of the ACM Multimedia Systems Conference (MMSys'11)*. 169–174.

Xi Liu, Florin Dobrian, Henry Milner, Junchen Jiang, Vyas Sekar, Ion Stoica, and Hui Zhang. 2012. A case for a coordinated Internet video control plane. In *Proceedings of the ACM SIGCOMM Conference (SIGCOMM'12)*. 359–370.

Yan Liu and Jack Y. B. Lee. 2014. On adaptive video streaming with predictable streaming performance. In *Proceedings of the IEEE Global Communications Conference (GLOBECOM'14)*. 1164–1169.

Thorsten Lohmar, Torbjörn Einarsson, Per Fröjdh, Frédéric Gabin, and Markus Kampmann. 2011. Dynamic adaptive HTTP streaming of live content. In *Proceedings of the 2011 IEEE International Symposium on a World of Wireless Mobile and Multimedia Networks (WoWMoM'11)*. 1–8.

Konstantin Miller, Savvas Argyropoulos, Nicola Corda, Alexander Raake, and Adam Wolisz. 2013. Optimal adaptation trajectories for block-request adaptive video streaming. In *Proceedings of the Packet Video Workshop*. 1–8.

Konstantin Miller, Dilip Bethanabhotla, Giuseppe Caire, and Adam Wolisz. 2015. A control-theoretic approach to adaptive video streaming in dense wireless networks. *IEEE Transactions on Multimedia* 17, 8, 1309–1322.

Konstantin Miller, Emanuele Quacchio, Gianluca Gennari, and Adam Wolisz. 2012. Adaptation algorithm for adaptive streaming over HTTP. In *Proceedings of the Packet Video Workshop*. 173–178.

Mariyam Mirza, Joel Sommers, Paul Barford, and Xiaojin Zhu. 2010. A machine learning approach to TCP throughput prediction. *IEEE/ACM Transactions on Networking* 18, 4, 1026–1039.

Ricky K. P. Mok, Xiapu Luo, Edmond W. W. Chan, and Rocky K. C. Chang. 2012. QDASH: A QoE-aware DASH system. In *Proceedings of the ACM Multimedia Systems Conference (MMSyS'12)*. 11–22.

MPEG. 2012. *MPEG-DASH (ISO/IEC 23009-1)*. MPEG.

J. Padhye, V. Firoiu, D. F. Towsley, and J. F. Kurose. 2000. Modeling TCP Reno performance: A simple model and its empirical validation. *IEEE/ACM Transactions on Networking* 8, 2, 133–145.

Toon De Pessemier, Katrien De Moor, Wout Joseph, Lieven De Marez, and Luc Martens. 2013. Quantifying the influence of rebuffering interruptions on the users quality of experience during mobile video watching. *IEEE Transactions on Broadcasting* 59, 1, 47–61.

Huynh-Thu Quan and Mohammed Ghanbari. 2008. Temporal aspect of perceived quality in mobile video broadcasting. *IEEE Transactions on Broadcasting* 54, 3, 641–651.

C. E. Rasmussen and C. K. I. Williams. 2006. *Gaussian Processes for Machine Learning*. MIT Press, Cambridge, MA.

Ulrich Reiter, Kjell Brunnström, Katrien De Moor, Mohamed-Chaker Larabi, Manuela Pereira, Antonio Pinheiro, Junyong You, and Andrej Zgank. 2014. Factors influencing quality of experience. In *Quality of Experience*, S. Moller and A. Raake (Eds.). Springer, 55–74.

Haakon Riiser, Tore Endestad, Paul Vigmostad, Carsten Griwodz, and Pal Halvorsen. 2012. Video streaming using a location-based bandwidth-lookup service for bitrate planning. *ACM Transactions on Multimedia Computing, Communications, and Applications* 8, 3, 1–19.

Michael Seufert, Sebastian Egger, Martin Slanina, Thomas Zinner, Tobias Hossfeld, and Phuoc Tran-Gia. 2014. A survey on quality of experience of HTTP adaptive streaming. *IEEE Communications Surveys and Tutorials* 17, 1, 469–492.

Kamal Deep Singh, Yassine Hadjadj-Aoul, and Gerardo Rubino. 2012. Quality of experience estimation for adaptive HTTP/TCP video streaming using H.264/AVC. In *Proceedings of the IEEE Consumer Communications and Networking Conference (CCNC'12)*. 127–131.

Iraj Sodagar. 2011. The MPEG-DASH standard for multimedia streaming over the Internet. *IEEE Multimedia* 18, 4, 62–67.

Wei Song and Dian W. Tjondronegoro. 2014. Acceptability-based QoE models for mobile video. *IEEE Transactions on Multimedia* 16, 3, 738–750.

Thomas Stockhammer. 2011. Dynamic adaptive streaming over HTTP—standards and design principles. In *Proceedings of the ACM Multimedia Systems Conference (MMSys'11)*. 133–144.

Gary J. Sullivan and Thomas Wiegand. 1998. Rate-distortion optimization for video compression. *IEEE Signal Processing Magazine* 15, 6, 74–90.

Viswanathan Swaminathan. 2013. Are we in the middle of a video streaming revolution? *ACM Transactions on Multimedia Computing, Communications, and Applications* 9, 1, 1–6.

Paul Sweeting. 2014. *Video in 2014: Going Live and Over the Top*. Research Report. GigaOM Media, San Francisco, CA.

Truong Cong Thang, Hung T. Le, Anh T. Pham, and Yong Man Ro. 2014. An evaluation of bitrate adaptation methods for HTTP live streaming. *IEEE Journal on Selected Areas in Communications* 32, 4, 693–705.

Guibin Tian and Yong Liu. 2012. Towards agile and smooth video adaptation in dynamic HTTP streaming. In *Proceedings of the ACM Conference on Emerging Networking Experiments and Technologies (CoNEXT'12)*. 109–120.

Sheng Wei and Viswanathan Swaminathan. 2014. Low latency live video streaming over HTTP 2.0. In *Proceedings of the Network and Operating System Support on Digital Audio and Video Workshop (NOSSDAV'14)*. 1–6.

Thomas Wiegand, Gary J. Sullivan, Gisle Bjontegaard, and Ajay Luthra. 2003. Overview of the H. 264/AVC video coding standard. *IEEE Transactions on Circuits and Systems for Video Technology* 13, 7, 560–576.

Xiaoqi Yin, Vyas Sekar, and Bruno Sinopoli. 2014. Toward a principled framework to design dynamic adaptive streaming algorithms over HTTP. In *Proceedings of the 13th ACM Workshop on Hot Topics in Networks (HotNets'14)*. 1–7.

Liu Yitong, Shen Yun, Mao Yinian, Liu Jing, Lin Qi, and Yang Dacheng. 2013. A study on quality of experience for adaptive streaming service. In *Proceedings of the IEEE International Conference on Communications Workshops (ICC'13)*. 682–686.

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

Chao Zhou, Xinggong Zhang, Longshe Huo, and Zongming Guo. 2012. A control-theoretic approach to rate adaptation for dynamic HTTP streaming. In *Proceedings of the IEEE Visual Communications and Image Processing Conference (VCIP'12)*. 1–6.

Xiaoqing Zhu, Zhi Li, Rong Pan, Joshua Gahm, and Rao Ru. 2013. Fixing multi-client oscillations in HTTP-based adaptive streaming: A control theoretic approach. In *Proceedings of the IEEE 15th International Workshop on Multimedia Signal Processing (MMSP'13)*. 230–235.

Xuan Kelvin Zou, Jeffrey Erman, Vijay Gopalakrishnan, Emir Halepovic, Rittwik Jana, Xin Jin, Jennifer Rexford, and Rakesh K. Sinha. 2015. Can accurate predictions improve video streaming in cellular networks? In *Proceedings of the 16th International Workshop on Mobile Computing Systems and Applications (HotMobile'15)*. 57–62.

Received March 2016; revised June 2016; accepted August 2016