TKN Telecommunication Networks Group

Technische Universität Berlin

Telecommunication Networks Group

QoE-Based Low-Delay Live Streaming Using Throughput Predictions*

Konstantin Miller, Abdel-Karim Al-Tamimi,

and Adam Wolisz

{miller,wolisz}@tkn.tu-berlin.de, altamimi@yu.edu.jo

Berlin, March 2016

TKN Technical Report TKN-16-001

TKN Technical Reports Series

Editor: Prof. Dr.-Ing. Adam Wolisz

* This technical report updates TR TKN-15-001.

Abstract

Recently, 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 varying network conditions in order to ensure a high quality of experience, 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 (**Lo**w-**L**atenc**y** **P**redicti**o**n-Based Ada**p**tation) that is designed to operate with a transport latency of few seconds. To reach this goal, LOLYPOP leverages TCP throughput predictions on multiple time scales, from 1 to 10 seconds, along with an estimate of the relative prediction error distribution. In addition to satisfying the latency constraint, the algorithm heuristically maximizes the quality of experience by maximizing the average video quality as a function of the number of skipped segments and quality transitions. In order 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 from the literature, called FESTIVE. We observed that the average video quality is by up to a factor of 3 higher than with FESTIVE. We also observed that LOLYPOP is able to reach a broader region in the quality of experience space, and thus it is better adjustable to the user profile or service provider requirements.

Contents

1. Introduction 4

2. Related Work 7

3. Considered System and its Model 11

4. LOLYPOP — Adaptation Algorithm for Low-Delay Live Streaming 14
4.1. Algorithm description 14
4.2. Tuning into the stream 15

5. TCP Throughput Traces 16

6. Short-Term TCP Throughput Prediction 18
6.1. Methodology 18
6.2. Prediction methods 19
6.3. Evaluation of the prediction accuracy 20
6.4. Estimating the relative prediction error 21
6.5. Estimating download success probabilities 22

7. Evaluation 23
7.1. Evaluation setting 23
7.2. Evaluation results 24

8. Conclusion 29

A. Information Contained in the Traces 36

B. Prediction Methods 37
B.1. Simple moving average 37
B.2. Linear extrapolation 37
B.3. Double exponential smoothing 37

C. Correlation of Underestimations and Overestimations 39

D. Fitting Prediction Error Distributions 40

Chapter 1.

Introduction

Over the last few years, we have been observing a dramatic change in video consumption patterns. The era of passive consumption of non-interactive "linear" content on a single device, the TV set, appears to be coming to an end, and a new mindset is being established: watch what I want, when I want, and where I want [13, 15]. 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 it will be video content [11].

Although the majority of streamed content is Video on Demand (VoD), the amount of live streaming is growing rapidly [58]. While current live streaming services can 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, a low latency 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 [23], while permanently or temporarily passive participants may be served with a delay of 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) [21]. Even though considerable effort has been put into developing networking architectures addressing this shortcoming [2, 7], 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 [14]. 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 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 Internet streaming: HTTP-Based Adaptive Streaming (HAS) [55]. One of the enablers of its success was the open standard MPEG-DASH (Dynamic Adaptive Streaming over HTTP) [44, 53]. The advantage of HAS comes from its usage of Hypertext Transfer Protocol (HTTP) leveraging an ubiquitous and highly optimized delivery infrastructure, including Content Delivery Networks (CDN's), caches, and proxies, 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 since 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 [22]. 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 **Lo**w-**L**atenc**y** **P**redicti**o**n-Based Ada**p**tation (LOLYPOP) that supports quality-based adaptation with a transport latency on the order of a few seconds. The approach introduced in LOLYPOP jointly considers three QoE components: the number of playback interruptions, the number of quality transitions, and the average video quality. Its goal is to maximize the average video quality as a function of the operating point defined by the other two components. The operating point is controlled by two input parameters: 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 download success probabilities for individual segments. To obtain these estimations, LOLYPOP leverages predictions of throughput distributions, computed from a time series prediction and an error estimation. We evaluate several time series prediction methods using TCP throughput traces collected in IEEE 802.11 Wireless Local Area Networks (WLAN's), 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. That is, 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 [25], 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 which is by up to a factor of 3 higher than the quality achieved by the baseline approach.

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

Chapter 2.

Related Work

In the last few years, adaptive streaming has been a very active research area. Even though the idea is not new [10, 27], it has recently moved into focus as the number of streaming video 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 [8] but, it was not until recently that it became the technology of choice for delivering video over the Internet [57, 32]. The adoption of the open standard MPEG-DASH (Dynamic Adaptive Streaming over HTTP) [44, 53] 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 [66, 67, 65, 63, 40], Markov decision processes [24, 6], machine learning [12], data mining [4], dynamic programming [33], data-driven techniques [36, 50, 17], and other heuristics selecting video quality based on the client's playback buffer level and average throughput [35, 43, 41, 34, 25]. Several studies use cross-layer information to improve client's performance [1], or jointly consider the problem of video quality selection and resource allocation on the lower layers of the Open Systems Interconnection (OSI) model [40, 28, 5]. Further studies propose to coordinate the quality selection process among the clients sharing a bottleneck link to allow for a more fair and efficient resource usage [16]. Finally, many popular streaming platforms such as YouTube[1] or 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, the results of individual studies are hard to compare with each other.

In contrast, 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 Over-the-Top (OTT) live streaming services [58] and a popular opinion that streaming over HTTP/TCP is less suitable for applications requiring a low delay. In [38], the authors 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 sub-second segment durations (approx. 31 kbps for one second segments). In [59], several adaptation algorithms are evaluated using buffer sizes from 6 to 20 seconds. The authors observe that the used set of representations can significantly affect the performance of indi-

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

vidual methods, resulting in by up to a factor of 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 trace. In [61], the authors 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 super-linear 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. In [30], an adaptation algorithm for live streaming is proposed 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% w.r.t. 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 [39, 68]. The evaluation in [39] reveals that the tested streaming clients can achieve a performance that is close to optimum w.r.t. the average media bit rate and number of playback interruptions, however, 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. In [43], the authors consider path probing techniques to obtain throughput estimations that, however, typically requires 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 in [37], 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 in order 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. Also, the maximum buffer size used in the evaluation is not specified. In [63], the authors 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. The study in [60] proposes 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. In [33], the authors 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. A heuristic adaptation algorithm is proposed in [29] 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 by 3% (17%) higher, the average quality transition magnitude is by 27% (9%) lower, and the number of quality transitions by 80% higher (46% lower) as compared to the two baseline approaches.

Since having accurate throughput predictions is beneficial for a number of applications, there are dedicated studies that address this subject, see [18] 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 [45, 18]. The authors in [18] observed that time series prediction methods perform quite well on the time scale of 50 seconds (Root Mean Square Relative Error (RMSRE) is less than 0.4 for about 90% of studied traces), but the accuracy strongly varies across studied network paths. 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 [45, 18, 42]. 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 [45, 42]. In addition, some analytical models target specific TCP flavors making their performance uncertain given recently developed variants [45].

Identifying performance goals for adaptive video streaming and expressing them in a way that facilitates objective measurements is an extremely challenging task. It must take into account human perception and cognitive processing — phenomena influenced by a hard to measure factors. The notion of Quality of Experience (QoE) was introduced in an effort to assess these phenomena and to help make them accessible to an objective evaluation process [22]. The number of factors influencing QoE is immense, and many of them have a high level of subjectivity that results in extremely complex modeling [49]. QoE for adaptive video streaming is an important and a fast developing research area [3, 51, 49, 54]. Rebuffering, initial delay, and quality fluctuations are factors that have not been part of traditional QoE metrics for video, but that have a tremendous impact on 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 [14]. Users are willing to accept a higher initial delay and higher video distortion, if it helps minimize playback interruptions [51, 20, 47, 52]. On the other hand, it was observed that video quality fluctuations resulting from dynamically changing the representation can have a negative impact on QoE [31, 64]. In particular, some studies conclude that a lower average video quality might be tolerated if it helps reduce the amount of representation transitions [46]. User engagement is another important metric, which is especially of interest for content providers because it directly relates to advertising-based revenue schemes [4, 14].

Chapter 3.

Considered System and its Model

In a live streaming system, the video content is recorded and published while streaming, 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. In order 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 transport latency variations caused by 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 nodes, etc.), transport latency (downloading the content), and client-side processing (demultiplexing, decoding, rendering).

In an HAS system, the video content is encoded in several representations varying w.r.t. their media characteristics such as spatial resolution, frame rate, compression rate, etc. They are configured by the service provider during the planning phase [51]. 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 playback 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 as well as 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 shutdown 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 in order 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, while the closed connection is replaced by a new one.

In addition to satisfying latency requirements, one of the main adaptation goals is to

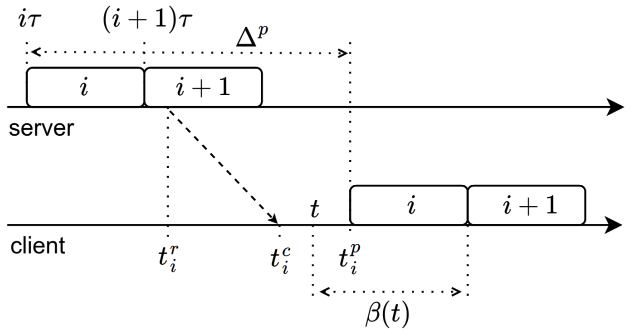

Figure 3.1.: Illustration of the used notation

maximize QoE. In the following, we define QoE as the triplet (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 [56]. 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 pair: 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), both 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, the provider's business model, etc.

In the following, we will introduce the notation used throughout the paper. All time-related variables are real-valued and represent continuous time. The start time of the recorded content is $t = 0$. The duration of video content contained in one segment is constant and denoted by τ. We use index $i \in \{0, 1, \ldots, n-1\}$ to indicate a particular segment in a stream. Segment i contains video material covering the time period $[i\tau, (i+1)\tau]$ and becomes available for download at time $(i+1)\tau$.

We denote the set of available video representations by $\mathcal{R}$, indexed by $j \in \{0, 1, \ldots, |\mathcal{R}| - 1\}$. We denote the size of a segment in bits by s_{ij}, and by $\bar{r}_{ij} = s_{ij}/\tau$, the Mean Media Bit Rate (MMBR) of segment i from representation j. We denote by $\bar{r}_j = 1/n \sum_{i=1}^{n} \bar{r}_{ij}$ the MMBR of representation j. We denote the size of a downloaded segment i by s_i, from the representation that was used to download it. Consequently, $\bar{r}_i = s_i/\tau$ denotes the MMBR of a downloaded segment i.

The time when the request to download segment i is sent 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 upper bound on the live latency by Δ^p. Consequently, the playback deadline of segment i is $t_i^p = i\tau + \Delta^p$. The value of Δ^p bounds the maximum transport latency, which is given by $\Delta^p - \tau$ if other latency components are neglected. Note that the maximum transport latency for individual segments can be smaller since it depends on the playback buffer level at the time of the request. The playback buffer level at time t is 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 is thus given by $0 < \beta(t_i^r) \leq \Delta^p - \tau$. See Figure 3.1 for an illustration.

We denote the fraction of segments skipped until 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 time t is denoted by $\Omega(t) \in [0,1]$.

Note that the traffic generated by a live streaming client might contain inter-request periods during which the client is waiting for the next segment to become available. When computing average application layer throughput, we exclude the inter-request periods in order to obtain an estimate of the throughput that was actually achieved during 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

$$\rho(t_1, t_2) = \frac{\sum_{i=l_1}^{l_2} s_i \cdot |[t_i^r, t_i^c] \cap [t_1, t_2]|}{\sum_{i=l_1}^{l_2} |[t_i^r, t_i^c] \cap [t_1, t_2]|}, \tag{3.1}$$

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, while 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.

Chapter 4.

LOLYPOP — Adaptation Algorithm for Low-Delay Live Streaming

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

4.1. Algorithm description

As described in Chapter 3, the goal of LOLYPOP is to maximize the average video quality as a function of the operating point, defined by the pair (Σ, Ω). The input parameters controlling the reached operating point are $\Sigma^* \in [0, 1]$, which controls the fraction of skipped segments (we will describe it in more details in the following), and $\Omega^* \in [0, 1]$, which is an upper bound on the (relative) number of quality transitions. 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. Computation of P^p_{ij} is described in detail in Section 6.5. For now, we assume that P^p_{ij} is given.

Let us consider the decision about downloading the segment i. 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 the download success probabilities cannot be computed (e.g., because the streaming session has just started or after a period of zero throughput), 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 in order to prevent $\Omega(t)$ from further exceeding the upper bound Ω^*. The pseudocode for the described algorithm is presented in Figure 4.1.

The intuition for letting LOLYPOP switch to a lower representation, even if the upper bound on the quality transitions is exceeded, is that preventing 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 video quality than to let the streaming stall [14].

```
Input: $t_i^r,\ t_i^p,\ \Sigma^*,\ \Omega^*$            ▷ Invocation time, playback deadline, config. parameters
Input: $\Omega\left(t_i^r\right)$            ▷ Current value for the relative number of quality transitions
Input: $\left(P_{ij}^p,\ j \in \{0,\dots,|\mathcal{R}|-1\}\right)$            ▷ Estimated download success probabilities, or -1
Input: $j^{\leftarrow} \in \{0,\dots,|\mathcal{R}|-1\}$            ▷ Repr. of the last successfully downloaded segment
Output: $j^*$            ▷ Selected representation
Require: $(i+1)\tau \le t_i^r < t_i^p \le t_i^r + T_{\max}$            ▷ Segment i available, playback deadline not passed and within prediction horizon
1: if $P_{ij}^p = -1,\ \forall j \in \{0,\dots,|\mathcal{R}|-1\}$ then            ▷ No estimation available
2:     $j^* = 0$            ▷ Select lowest representation
3: else            ▷ An estimation of download success probabilities is available
4:     $j' = \max\left(\{0\} \cup \left\{j \in \{0,\dots,|\mathcal{R}|-1\} \mid 1 - P_{ij}^p \le \Sigma^*\right\}\right)$            ▷ Max. representation satisfying $\Sigma^*$, 0 if none
5:     if $\Omega\left(t_i^r\right) \le \Omega^*$ then            ▷ Transition to a higher representation is possible
6:         $j^* = j'$
7:     else            ▷ Transition to a higher representation is not possible
8:         $j^* = \min\left(j', j^{\leftarrow}\right)$
```

Figure 4.1.: Pseudocode of LOLYPOP

4.2. Tuning into the stream

When the client is about to start a new streaming session, it has to decide which segment to download first and in which representation. The client might start with the newest segment among those available for download, maximizing the probability that the first segment will be downloaded before its playback deadline but increasing the initial delay. In contrast, taking the oldest 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 and selects the first segment i_0 as the oldest segment whose playback deadline is at least τ seconds into the future: $i_0 = \min\{i \ge 0 \mid (i+1)\tau \le t \wedge t_i^p \ge t + \tau\}$, where t is the time when a user tunes into the stream. Furthermore, it downloads the first segment in the lowest quality in order to minimize the risk of missing its playback deadline and thus unnecessarily increasing the initial delay. Due to the small segment duration, the low quality of the first segment has negligible impact on the average video quality of the streaming session. The intuition is that the available network resources should at least support the download of a segment in its lowest quality in less time than the segment duration. If the first segment can be downloaded before its playback deadline, as expected, the start-up delay will thus lie in the interval $[\tau,\ \Delta^p - \tau]$, which can be seen by transforming the equation for i_0, using $t_i^p = i\tau + \Delta^p$. LOLYPOP applies the same decision process when the client skips a segment and has to select a segment to proceed with.

Chapter 5.

TCP Throughput Traces

As previously stated, LOLYPOP leverages estimations of probabilities P_{ij}^{p} that segment i can be downloaded in representation j before its playback deadline. In order to develop an efficient estimator, it requires a data set, that is, a collection of TCP throughput traces from IEEE 802.11 WLAN's, to evaluate the accuracy and error distributions of different time-series prediction methods. In addition, such a data set is required to evaluate the proposed adaptation algorithm under different network conditions. Due to the targeted transport latency of a few seconds, the required data set must contain throughput averages computed over relatively small time intervals: 1 second or less. To the best of our knowledge, there exists no such publicly available data set. 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 include public hotspots (indoor and outdoor), campus hotspots, and access points in residential environments. The traces were collected using laptops running Ubuntu 13.04 and Ubuntu 14.04 operating systems with default Media Access Control (MAC) and TCP configurations.

We collected 127 traces of continuous downstream TCP flows, lasting between 600 and 3600 seconds each. In order to focus on the more challenging scenarios, we removed traces with a Coefficient of Variation (CV) of less than 0.1 resulting in 92 traces with a total length of 45 hours. As a sender, we used either a server located at the TU Berlin campus (running Ubuntu 12.04) or an Amazon EC2 micro instance located in Ireland (running Ubuntu 14.04). More information about the traces, as well as an example figure depicting the throughput of one complete trace, are provided in Appendix A. All traces are available upon request.

Figure 5.1 provides an illustration of throughput variability and temporal correlation that are among the main factors affecting predictability. The figure shows boxplots for selected sampling intervals of the mean throughput, Coefficient of Variation (CV), auto-correlation at lag 1, and auto-correlation after differencing at lag 1. The CV is defined as the standard deviation divided by the mean. Auto-correlation at lag 1 shows how probable it is that a large throughput value is followed by another large value (auto-correlation close to 1) or a small value (auto-correlation close to -1). A value close to 0 indicates no temporal correlation between subsequent values. Auto-correlation after differencing quantifies the correlation of throughput changes. The computed statistics confirm that our traces cover a broad range of network conditions. 50% of the traces have a mean throughput between 4 and 11 Mbps, while for 90% of traces the mean lies between approximately 1 and 13 Mbps. 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 auto-correlation values of over 0.6 at a sampling interval of 2 seconds, while 95% still have auto-correlation values over 0.3, indicating

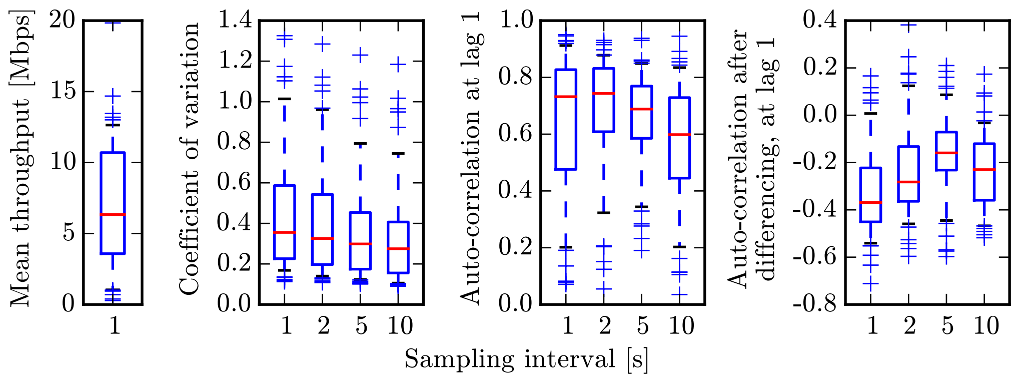

Figure 5.1.: Trace statistics: mean throughput (equal for all sampling intervals), Coefficient of Variation (CV), auto-correlation at lag 1, and auto-correlation, after differencing, at lag 1. Horizontal line: median, box: quartiles, whiskers: 0.5 and 0.95 quantiles, flier points: outliers.

significant temporal correlation between subsequent measurements. At the same time, the time series of throughput changes exhibits a strong negative auto-correlation, indicating that a throughput increase is likely to be followed by a throughput decrease.

Chapter 6. Short-Term TCP Throughput Prediction

In this chapter, we present our approach to estimating 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 $\rho\,(t - iT, t - (i-1)T)$, $i \in \{1, \ldots, n\}$, it computes predictions $\hat{\rho}\,(t + (i-1)T, t + iT)$, $i \in \{1, \ldots, k\}$, where T is the averaging interval.

As described in Chapter 3, the upper bound on the download duration $t_i^p - t_i^r$ for segment i takes values from the range $(0, \Delta^p - \tau]$, 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, always 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

$$\epsilon\,(t_1,\, t_2) = \frac{\left|\max\left(\hat{\rho}\,(t_1,\, t_2)\,,\, \rho_{\min}\right) - \max\left(\rho\,(t_1,\, t_2)\,,\, \rho_{\min}\right)\right|}{\max\left(\rho\,(t_1,\, t_2)\,,\, \rho_{\min}\right)}\,. \tag{6.1}$$

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.

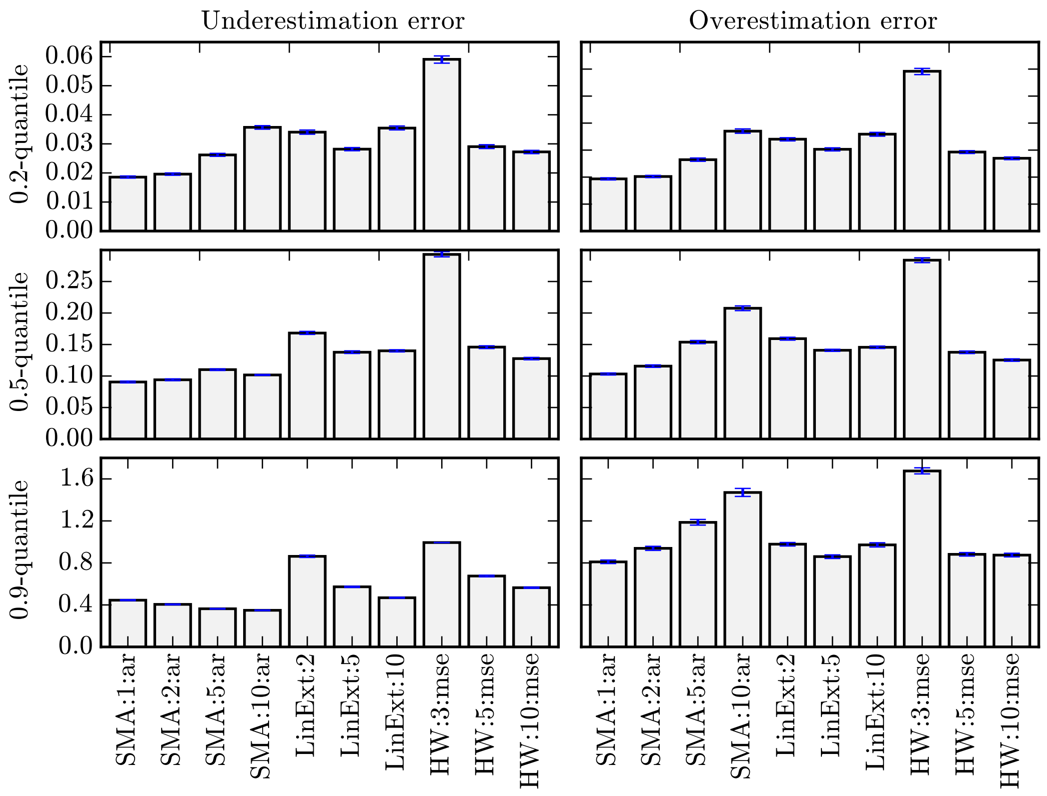

Figure 6.1.: Relative prediction error quantiles for the complete data set, on the time scale of 5 seconds. Left column shows the underestimation error, right column the overestimation error. The error bars show confidence intervals for the confidence level of 0.95. See Section 6.3 for details.

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 Integrated Moving Average (ARIMA), and several machine learning based methods [9]. Our throughput prediction results will focus on three simple methods: SMA, linear extrapolation, and double exponential smoothing (Holt-Winters). A brief description of these methods is provided in Appendix B. We abbreviate the methods by ⟨type⟩:⟨n⟩:⟨parameters⟩, where ⟨type⟩ is the name of the method, n is the number of past throughput values used as input, and ⟨parameters⟩ include further optional configuration parameters. For example, SMA:n:ar denotes SMA with arithmetic mean, and HW:n:mse denotes Holt-Winters with Mean Squared Error (MSE) used 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 of the traces to identify the method that performs best over a broad range of network environments. In the second step, we evaluate how the prediction accuracy varies over individual traces.

To compare the prediction accuracy over the complete data set, we computed the 0.2-quantiles, 0.5-quantiles, and 0.9-quantiles of the prediction error. For example, a 0.2-quantile of 0.3 means that in 20% of all collected data points, the relative prediction 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 over 0.8. The results for the sampling interval of 5 seconds are shown in Figure 6.1. 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%, while only less than 10% of data points have an overestimation error of 80% and larger. This is somewhat surprising since SMA:1:ar is the most naïve method that uses only 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 5.1. Also, methods using a small number of history points implicitly detect level shifts and do not propagate outliers. These two issues were reported to be known challenges in TCP throughput prediction [18].

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 Functions (ECDF's) of these fractions over individual traces for the sampling interval of 5 seconds, is shown in Figure 6.2. The first/second/third column shows for each trace the fraction of measurements with a relative error below 0.2/0.3/0.5 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 worse performance, while points above (y-value over 0.3) correspond to traces with better performance. The fact that the graph passes through point $(0.8, 0.3)$ means that in 70% of traces (sampled at 5 s), less than 20% of the overestimations have a relative error of 0.5 or more.

From Figure 6.2, we observe that the prediction strongly varies across traces. There are traces, where 90% of overestimations have an error less than 20%, while 100% of predictions have an error less than 50%. A video client might account for a relative error of this magnitude by using a fixed safety margin, that is, by always selecting a media bit rate which 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%, while more than 40% of the overestimations still have an error greater than 100%. Setting a high fixed safety margin to account for such "bad" traces would result in significant underutilization of network resources, lower media bit rate and thus lower QoE in the "well-behaving" traces. On the other hand,

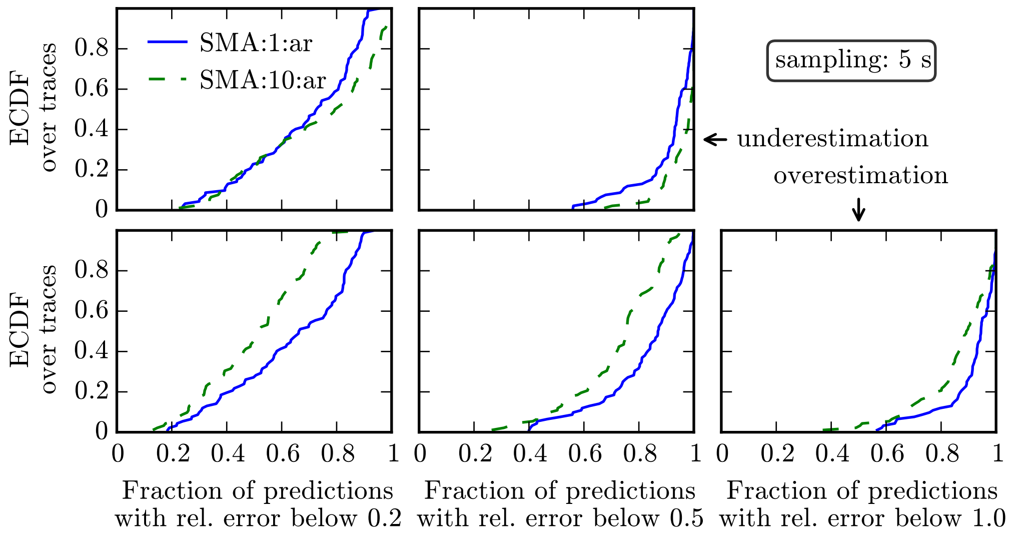

Figure 6.2.: Performance of SMA:1:ar and SMA:10:ar for individual traces. See Section 6.3 for details.

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 are well balanced on all time scales. Both occur in approximately 50% ± 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 over 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 5.1. The distribution of per-trace values is depicted in Appendix C.

6.4. Estimating the relative prediction error

In order to estimate the download success probability, it is not sufficient to perform a time series prediction because the uncertainty of such a prediction can be quite high (as shown previously) and because it can vary across different network environments. Although there are approaches that allow to explicitly predict a distribution such as Gaussian process method [48], 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 to account for the long-term non-stationarities by discarding values whose age exceeds a certain threshold. Another approach is to select a distribution type and to fit its parameters dynamically from the data. In our evaluation, we will use the former method, since it results in good performance and since with the latter method the computation of the model parameters involves an optimization step, which is resource-consuming. In Appendix D we present our results on fitting several well-known distribution types to relative prediction errors: exponential, normal, logistic, and Lomax (shifted Pareto) [26]. We observe that the Lomax distribution provides the best fit. We therefore recommend to use the Lomax distribution to model prediction errors when evaluating adaptation approaches with synthetic data, as done in [63], for example.

6.5. Estimating download success probabilities

In this section, we describe our approach to estimating download success probabilities $\left(P^p_{ij}, j \in \{0, \ldots, |\mathcal{R}| - 1\}\right)$ that at time t^r_i s_{ij} bytes can be downloaded in representation j before its playback deadline t^p_i. We denote by $T_{\max} \in \mathbb{N}$ the maximum prediction horizon in seconds. Consequently, at time $t \in \mathbb{N}$, the client computes the average application layer throughput (defined in (3.1)) for time intervals $[t - T, t]$ for $T \in \{1, \ldots, T_{\max}\}$, followed by computing throughput predictions for time intervals $[t, t + T]$. If the throughput cannot be computed, no prediction is provided either. Finally, the client computes the relative prediction error for the interval $[t - T, t]$ as

$$\tilde{\epsilon}\left(t - T, t\right) = \frac{\max\left(\hat{\rho}\left(t - T, t\right), \rho_{\min}\right) - \max\left(\rho\left(t - T, t\right), \rho_{\min}\right)}{\max\left(\rho\left(t - T, t\right), \rho_{\min}\right)} \,. \tag{6.2}$$

In contrast to the definition in (6.1), $\tilde{\epsilon}\left(t - T, t\right) \in (-1, \infty)$ is defined without taking the absolute value for the purposes of presentation.

We assume that predictions are computed every second, so that at time t^r_i, the most recent predictions were computed at time $\lfloor t^r_i \rfloor$. In order 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 $[\lfloor t^r_i \rfloor, \lceil t^p_i \rceil]$ since, due to the distribution of inter-request delays or due to a throughput outage, a prediction for this time interval might not be available. Let t^π_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 $\hat{\rho}_i = \hat{\rho}\left(t^\pi_i,\, t^\pi_i + T\right)$ be the corresponding throughput prediction. $\epsilon_i = \epsilon\left(t^\pi_i,\, t^\pi_i + T\right)$ and $\tilde{\epsilon}_i = \tilde{\epsilon}\left(t^\pi_i,\, t^\pi_i + T\right)$ shall denote the relative prediction errors, as defined in (6.1) and (6.2). Further, we denote by $\Phi^u_i\left(\epsilon_i\right)$ and $\Phi^o_i\left(\epsilon_i\right)$ 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 for $\tilde{\epsilon}_i$ is given by

$$\Phi_i\left(\tilde{\epsilon}_i\right) = \begin{cases} P^u_i \cdot \Phi^u_i\left(\epsilon_i\right) & \text{for } \tilde{\epsilon}_i < 0 \\ P^u_i + \left(1 - P^u_i\right) \cdot \Phi^o_i\left(\epsilon_i\right) & \text{otherwise}\,. \end{cases} \tag{6.3}$$

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

$$P^p_{ij} = P\left[\frac{s_{ij}}{t^p_i - t^r_i} \leq \frac{\hat{\rho}_i}{1 + \tilde{\epsilon}_i}\right] = \Phi_i\left(\frac{\hat{\rho}_i\left(t^p_i - t^r_i\right)}{s_{ij}} - 1\right) \,. \tag{6.4}$$

Chapter 7. Evaluation

We evaluated the performance of LOLYPOP using our collected throughput traces and comparing it against the state-of-the-art algorithm FESTIVE [25] 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 written in Python[1]. We equipped the developed prototype 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[2] as video content, which is an animated movie of approximately 10 minutes duration. This video was selected due to the availability of raw video data, allowing us to generate representations with high MMBR's. We encoded 9 representations with MMBR's distributed between 100 and 20000 kbps with exponentially increasing intervals: 101, 194, 377, 730, 1415, 2743, 5319, 10314, and 20000 kbps, using the H.264/MPEG-4 AVC [62] compression format. The chosen intervals correspond to a roughly linear increase of the video quality in terms of Peak Signal-to-Noise Ratio (PSNR) [56]. The encoding was performed using the avconv[3] utility using two passes, with a configuration targeting at low MMBR variations among individual segments.

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 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 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^*.

[1]http://www.python.org
[2]http://www.bigbuckbunny.org
[3]http://libav.org/avconv.html

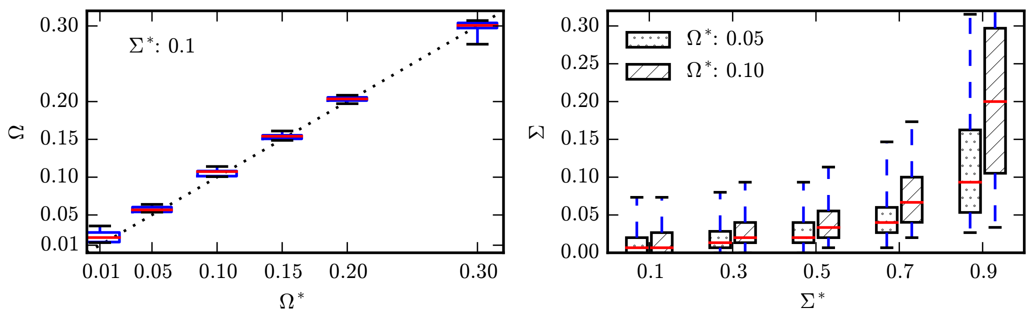

Figure 7.1.: Ω as function of Ω^* (left), and Σ as function of Σ^* (right), for LOLYPOP. The distributions over the traces are shown as boxplots, where the horizontal line represents the median, the box represents the quartiles, and the whiskers represent the 0.05 and 0.95 quantiles.

The FESTIVE adaptation algorithm was evaluated with a broad range of values around the default configuration in [25]. 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), 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 2880 configurations. We disabled the randomizer feature of FESTIVE since 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 since we considered it as one of its core features. Finally, we would like to point out that FESTIVE bases its decisions upon the knowledge of the MMBR of a representation, while 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, to explore the region of reachable operating points, and to evaluate the average video quality as a function of the operating point.

First, we study the dependency of the reached operating point (Σ, Ω) on the input parameters Σ^* and Ω^*. Figure 7.1 (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 are 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. Also note that a value of $\Omega = 0.01$ means that during the whole streaming session, there are only 3 quality transition, which, in a wireless network, is an extremely low value. Figure 7.1 (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

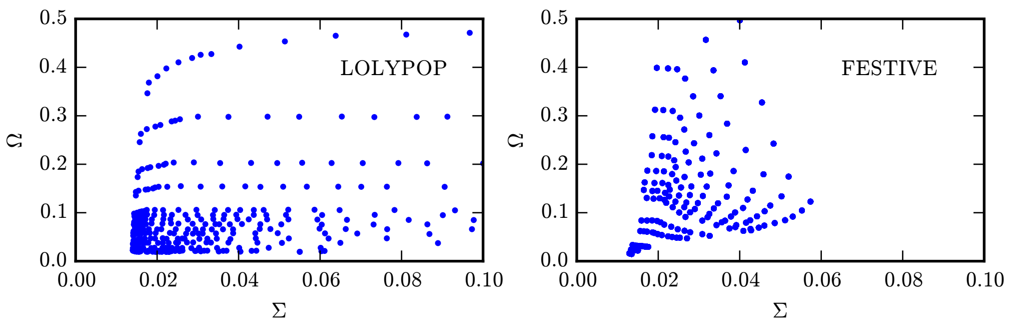

Figure 7.2.: Scatter plots of covered (Σ, Ω) regions for LOLYPOP (left), and FESTIVE (right).

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 this region the more flexible the algorithm can be tuned to the QoE requirements defined for a streaming session. Figure 7.2 shows scatter plots of achieved (Σ, Ω) values for LOLYPOP (left) and FESTIVE (right). We observe that the studied LOLYPOP configurations cover a broader range of (Σ, Ω) values and thus enable 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 since it might be achieved by selecting an unnecessary 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 durations contained in several traces. In order 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 have a Σ below 0.01, further 10 below 0.05, further 7 below 0.1, one had 0.12 and one has the highest Σ of 0.15.

As the main result of our evaluation, we would like to characterize the average video quality as a function of the operating point. Figure 7.3 visualizes the average video quality as a function of quality transitions for different numbers of skipped segments. For different values of Σ (on the x-axis), we first determine the configuration that (i) achieves the highest average video quality over all traces, (ii) whose average fraction of skipped segments is less or equal to Σ, and (iii) whose average (relative) number of quality transitions is less then 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 7.3.

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 3 times higher average quality. For high numbers of quality transitions, the difference slightly increases with the number of skipped segments. 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

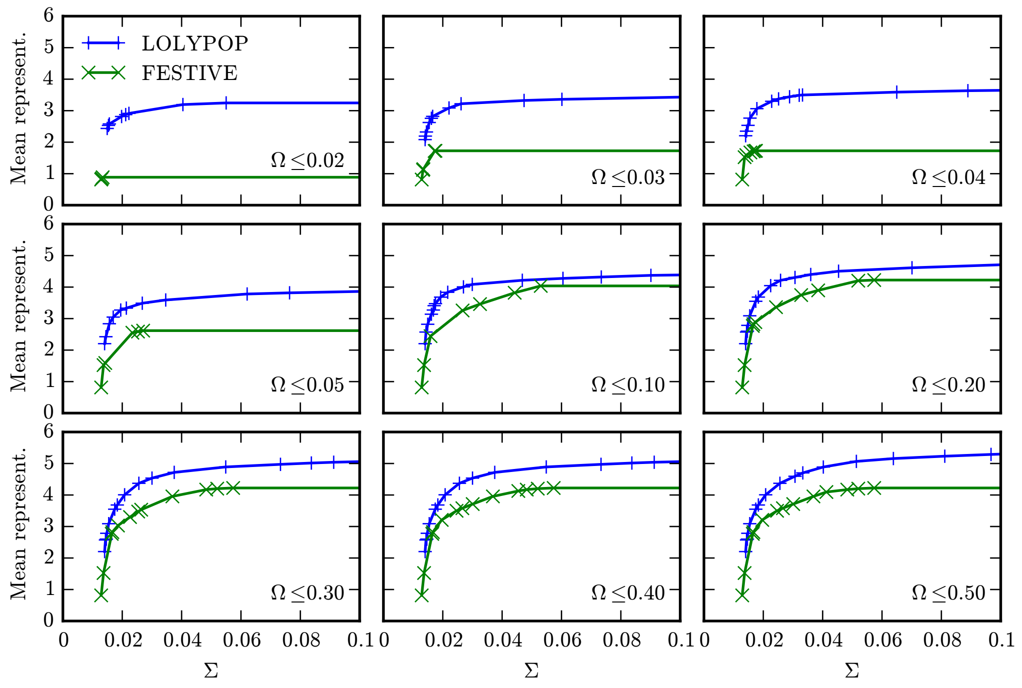

Figure 7.3.: Average video quality (representation) as a function of the number of skipped segments Σ for different numbers of quality transitions Ω.

video quality. In the evaluated network environments, the "knee" is typically slightly below $\Sigma = 0.02$. In other words, accepting 0.5 to 1 percent 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 $\Omega \leq 0.2$).

While Figure 7.3 shows mean values over all 92 traces, we generated similar plots for each trace individually. 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, while there existed no trace where all 9 curves were pointwise greater for FESTIVE. Furthermore, in order 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 for $\Omega = 0.2$, in 53% of traces, LOLYPOP had a higher integral than FESTIVE; in 38% of traces, FESTIVE had a higher integral; and in the remaining traces, the values were equal. The corresponding values for $\Omega \in \{0.03, 0.04, 0.05, 0.1, 0.2, 0.3, 0.4, 0.5\}$ are (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 7.3, Figure 7.4 presents plots of quality vs. quality transitions for different levels of skipped segments. Here, we observe a similar situtation, 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.

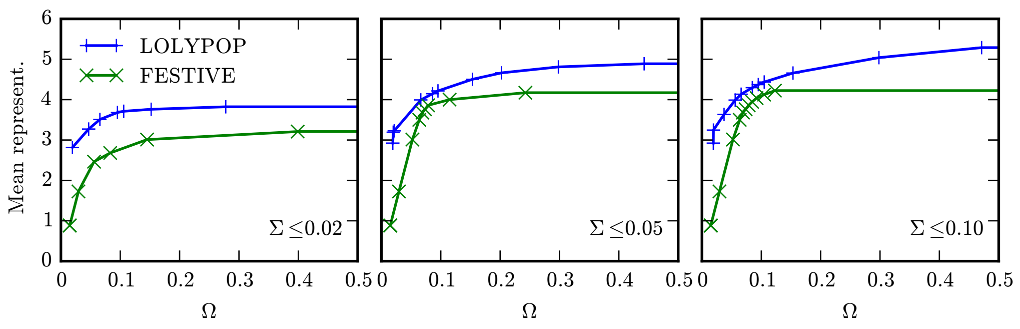

Figure 7.4.: Average video quality (representation) as a function of the number of quality transitions Ω for different numbers of skipped segments Σ.

Finally, Figure 7.5 depicts four example runs illustrating the behavior of the proposed algorithm with different configurations. For each run, three plots are shown. The top one depicts network throughput and segment MMBR. The middle one depicts the representations selected for individual segments and the mean value. 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, since the whole streaming session has less than 1000 segments. We observe that the algorithm reacts not only to decreased throughput, as seen between seconds 30 and 50, but also to increased uncertainty in throughput dynamics as seen after the strong downward fluctuation at second 170. A higher Σ^*, as seen in the top right subplots, results in a more aggressive behavior accepting a higher probability for skipping a segment and a higher average quality. The two sets of subplots at the bottom depict runs with $\Omega^* = 0.1$. The algorithm is allowed to have more quality transitions, resulting in a further improvement of the average video quality.

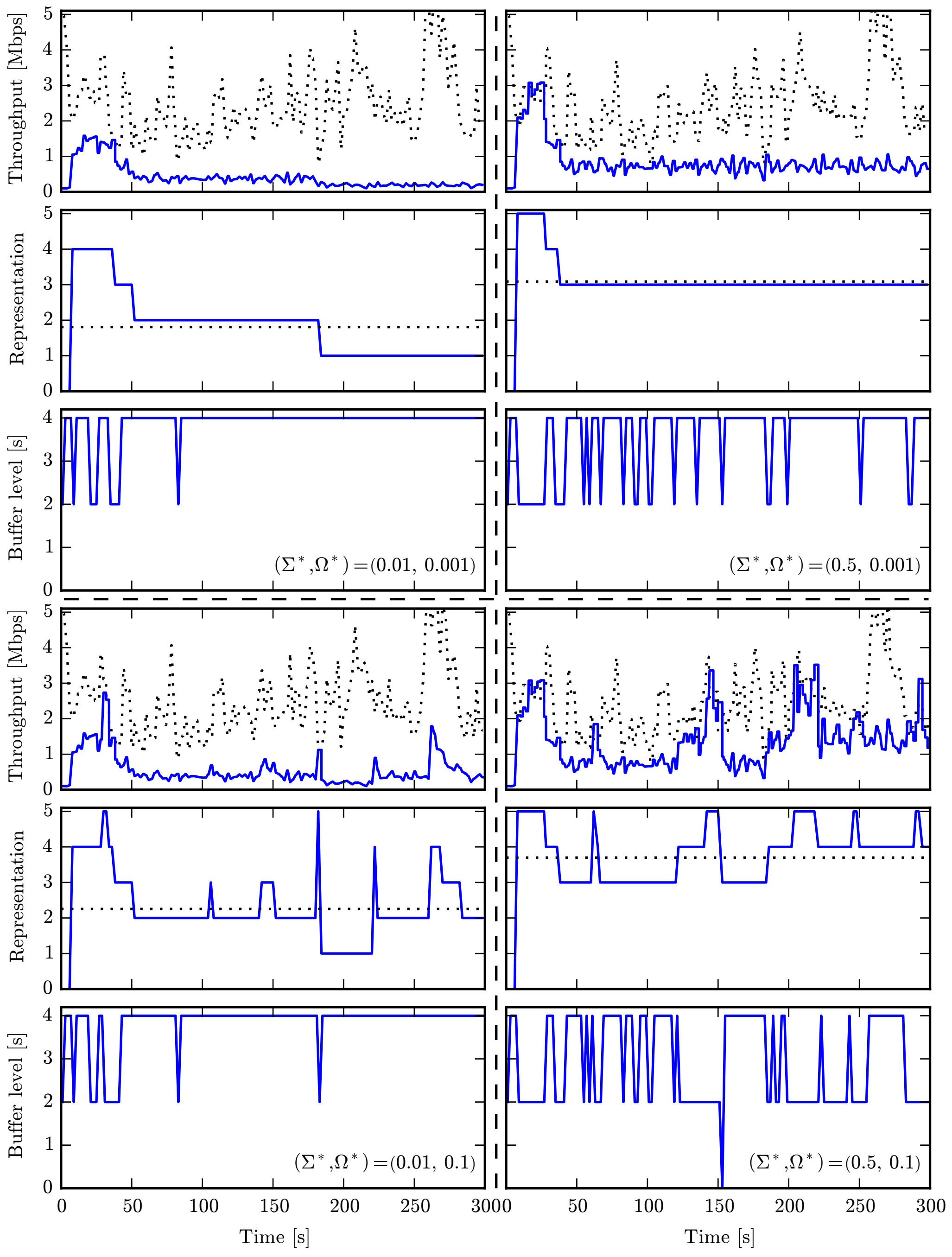

Figure 7.5.: Four example runs with different algorithm configurations. See Section 7.2 for details.

Chapter 8. Conclusion

In the presented study, we addressed the problem of maximizing QoE for HTTP-based adaptive low-delay live 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 predictions of TCP throughput distributions on time scales from 1 to 10 seconds. We studied the performance of several time series prediction methods using IEEE 802.11 traces from various network environments. We observed that the most naïve 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 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 collected throughput traces. We observed that LOLYPOP is able to reach a broad range of operating points, and to outperform the baseline approach w.r.t. 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, in particular Σ^*, in order to achieve a target level of skipped segments, and (ii) to further improve prediction accuracy. Furthermore, we evaluate how Δ^p can be dynamically tuned in order to achieve minimum latency without dropping the QoE. Finally, we plan to evaluate the potential to further reduce latency by reducing the video segment duration.

Bibliography

[1] Abdallah S Abdallah and Allen B Mackenzie. A Cross-Layer Controller for Adaptive Video Streaming over IEEE 802.11 Networks. In *Proc. of IEEE International Conference on Communications (ICC)*, pages 6797–6802, London, UK, 2015.

[2] Cristina Aurrecoechea, Andrew T. Campbell, and Linda Hauw. A Survey of QoS Architectures. *Multimedia Systems*, 6(3):138–151, 1998.

[3] Athula Balachandran, Vyas Sekar, Aditya Akella, Srinivasan Seshan, Ion Stoica, and Hui Zhang. A Quest for an Internet Video Quality-of-Experience Metric. In *Proc. of ACM Workshop on Hot Topics in Networks (HotNets)*, pages 97–102, Redmond, WA, 2012.

[4] Athula Balachandran, Vyas Sekar, Aditya Akella, Srinivasan Seshan, Ion Stoica, and Hui Zhang. Developing a Predictive Model of Quality of Experience for Internet Video. In *Proc. of ACM SIGCOMM*, pages 339–350, Hong Kong, 2013.

[5] Dilip Bethanabhotla, Giuseppe Caire, and Michael J Neely. Adaptive Video Streaming for Wireless Networks With Multiple Users and Helpers. *IEEE Trans. on Communications*, 63(1):268–285, 2015.

[6] Ayub Bokani, Mahbub Hassan, and Salil Kanhere. HTTP-Based Adaptive Streaming for Mobile Clients using Markov Decision Process. In *In Proc. of Intl. Packet Video Workshop (PV)*, San Jose, CA, 2013.

[7] Jorge Carapinha, Roland Bless, Christoph Werle, Konstantin Miller, Virgil Dobrota, Andrei Bogdan Rus, Heidrun Grob-Lipski, and Horst Roessler. Quality of Service in the Future Internet. In *Proc. of ITU-T Kaleidoscope*, Pune, India, 2010.

[8] Sharon Carmel, Tzur Daboosh, Eli Reifman, Naftali Shani, Ziv Eliraz, Dror Ginsberg, Edan Ayal, and Kfar Saba. Network Media Streaming, 2002. Patent No. US 6389473, Filed March 24, 1999, Issued May 14, 2002.

[9] Christopher Chatfield. *The Analysis of Time Series: an Introduction.* Taylor & Francis, Abingdon, UK, 6th. edition, 2003.

[10] Zhigang Chen, See-Mong Tan, Roy H. Campbell, and Yongcheng Li. Real Time Video and Audio in the World Wide Web. In *In Proc. of International World Wide Web Conference*, Boston, MA, USA, 1995.

[11] Cisco. Cisco Visual Networking Index: Forecast and Methodology, 2013 - 2018. White Paper, Cisco Systems, Inc., San Jose, CA, 2014.

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

[13] ComScore. U.S. Digital Future in Focus. White Paper, comScore, Inc., Reston, VA, 2014.

[14] Conviva. Viewer Experience Report. White Paper, Conviva, Foster City, CA, 2014.

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

[16] A El Essaili, D Schroeder, D Staehle, M Shehada, W Kellerer, and E Steinbach. Quality-of-Experience driven Adaptive HTTP Media Delivery. In *Proc. IEEE International Conference on Communications (ICC)*, pages 2480–2485, Budapest, Hungary, 2013.

[17] Jia Hao, Roger Zimmermann, and Haiyang Ma. GTube: Geo-Predictive Video Streaming over HTTP in Mobile Environments. In *Proc. of ACM Multimedia Systems Conference (MMSys)*, pages 259–270, Singapore, 2014.

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

[19] Myles Hollander, Douglas A. Wolfe, and Eric Chicken. *Nonparametric Statistical Methods*. Wiley, New York, NY, 3rd. edition, 2014.

[20] T. Hossfeld, Sebastian Egger, Raimund Schatz, Markus Fiedler, Kathrin Masuch, and C. Lorentzen. Initial Delay vs. Interruptions: Between the Devil and the Deep Blue Sea. In *Proc. of Workshop on Quality of Multimedia Experience (QoMEX)*, Yarra Valley, Australia, 2012.

[21] ITU-T. Definition of Terms Related to Quality of Service (ITU-T E.800). Recommendation, ITU-T, 2008.

[22] ITU-T. Vocabulary for Performance and Quality of Service, Amendment 2: New Definitions for Inclusion in Recommendation ITU-T P.10/G.100. Recommendation, ITU-T, 2008.

[23] ITU-T. Requirements for low-latency interactive multimedia streaming (ITU-T F.746.1). Recommendation, ITU-T, 2014.

[24] Dmitri Jarnikov and Tanr Özçelebi. Client Intelligence for Adaptive Streaming Solutions. *Signal Processing: Image Communication*, 26(7):378–389, 2011.

[25] Junchen Jiang, Vyas Sekar, and Hui Zhang. Improving Fairness, Efficiency, and Stability in HTTP-Based Adaptive Video Streaming With FESTIVE. *IEEE/ACM Trans. on Networking*, 22(1):326–340, 2014.

[26] Norman Lloyd Johnson, Samuel Kotz, and Narayanaswamy Balakrishnan. *Continuous Univariate Distributions*. Wiley, New York, NY, 2nd. edition, 1994.

[27] Hemant Kanakia, Partho P. Mishra, and Amy R. Reibman. An Adaptive Congestion Control Scheme for Real Time Packet Video Transport. *IEEE/ACM Transactions on Networking*, 3(6):671–682, 1995.

[28] Hieu Le, Arash Behboodi, and Adam Wolisz. Quality Driven Resource Allocation for Adaptive Video Streaming in OFDMA Uplink. In *Proc. IEEE 26th Annual International Symposium on Personal, Indoor, and Mobile Radio Communications (PIMRC)*, pages 1277–1282, Hong Kong, 2015.

[29] Hung T. Le, Duc V. Nguyen, Nam Pham Ngoc, Anh T. Pham, and Truong Cong Thang. Buffer-Based Bitrate Adaptation for Adaptive HTTP Streaming. In *Proc. International Conference on Advanced Technologies for Communications (ATC)*, pages 33–38, Ho Chi Minh City, Vietnam, 2013.

[30] Hung T Le, Hai N Nguyen, Nam Pham Ngoc, Anh T Pham, Hoa Le Minh, and Truong Cong Thang. Quality-Driven Bitrate Adaptation Method for HTTP Live-Streaming. In *Proc. IEEE International Conference on Communication Workshop (ICCW)*, pages 1771–1776, London, UK, 2015.

[31] Blazej Lewcio, Benjamin Belmudez, Theresa Enghardt, and Sebastian Möller. On the Way to High-Quality Video Calls in Future Mobile Networks. In *Proc. of International Workshop on Quality of Multimedia Experience (QoMEX)*, pages 43–48, Mechelen, Belgium, 2011.

[32] Baochun Li, Zhi Wang, Jiangchuan Liu, and Wenwu Zhu. Two Decades of Internet Video Streaming: A Retrospective View. *ACM Transactions on Multimedia Computing, Communications, and Applications*, 9(1s):1–20, 2013.

[33] Zhi Li, Ali C. Begen, Joshua Gahm, Yufeng Shan, Bruce Osler, and David Oran. Streaming Video over HTTP with Consistent Quality. In *Proc. of the 5th ACM Multimedia Systems Conference (MMSys)*, pages 248–258, New York, NY, 2014.

[34] Zhi Li, Xiaoqing Zhu, Josh Gahm, Rong Pan, Hao Hu, Ali C Begen, and Dave Oran. Probe and Adapt: Rate Adaptation for HTTP Video Streaming At Scale. *IEEE Journal on Selected Areas in Communications*, 32(4):719–733, 2014.

[35] Chenghao Liu, Imed Bouazizi, and Moncef Gabbouj. Rate Adaptation for Adaptive HTTP Streaming. In *Proc. of ACM Multimedia Systems Conference (MMSys)*, pages 169–174, San Jose, CA, USA, 2011.

[36] Xi Liu, Florin Dobrian, Henry Milner, Junchen Jiang, Vyas Sekar, Ion Stoica, and Hui Zhang. A Case for a Coordinated Internet Video Control Plane. In *Proc. of ACM SIGCOMM*, pages 359–370, Helsinki, Finland, 2012.

[37] Yan Liu and Jack Y. B. Lee. On Adaptive Video Streaming with Predictable Streaming Performance. In *Proc. of IEEE Global Communications Conference (GLOBECOM)*, pages 1164–1169, Austin, TX, 2014.

[38] Thorsten Lohmar, Torbjörn Einarsson, Per Fröjdh, Frédéric Gabin, and Markus Kampmann. Dynamic Adaptive HTTP Streaming of Live Content. In *Proc. of 2011 IEEE International Symposium on a World of Wireless Mobile and Multimedia Networks (WoWMoM)*, pages 1–8, Lucca, Italy, 2011.

[39] Konstantin Miller, Savvas Argyropoulos, Nicola Corda, Alexander Raake, and Adam Wolisz. Optimal Adaptation Trajectories for Block-Request Adaptive Video Streaming. In *Proc. of the Packet Video Workshop*, pages 1–8, San Jose, CA, 2013.

[40] Konstantin Miller, Dilip Bethanabhotla, Giuseppe Caire, and Adam Wolisz. A Control-Theoretic Approach to Adaptive Video Streaming in Dense Wireless Networks. *IEEE Transactions on Multimedia*, 17(8):1309 – 1322, 2015.

[41] Konstantin Miller, Emanuele Quacchio, Gianluca Gennari, and Adam Wolisz. Adaptation Algorithm for Adaptive Streaming over HTTP. In *Proc. of the Packet Video Workshop*, pages 173–178, Munich, Germany, 2012.

[42] Mariyam Mirza, Joel Sommers, Paul Barford, and Xiaojin Zhu. A Machine Learning Approach to TCP Throughput Prediction. *IEEE/ACM Transactions on Networking*, 18(4):1026–1039, 2010.

[43] Ricky K. P. Mok, Xiapu Luo, Edmond W. W. Chan, and Rocky K. C. Chang. QDASH: A QoE-Aware DASH System. In *Proc. of ACM Multimedia Systems Conference (MMSyS)*, pages 11–22, Chapel Hill, NC, 2012.

[44] MPEG. MPEG-DASH (ISO/IEC 23009-1). *Moving Picture Experts Group*, 2012.

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

[46] Toon De Pessemier, Katrien De Moor, Wout Joseph, Lieven De Marez, and Luc Martens. Quantifying the Influence of Rebuffering Interruptions on the Users Quality of Experience During Mobile Video Watching. *IEEE Transactions on Broadcasting*, 59(1):47–61, 2013.

[47] Huynh-Thu Quan and Mohammed Ghanbari. Temporal Aspect of Perceived Quality in Mobile Video Broadcasting. *IEEE Transactions on Broadcasting*, 54(3):641–651, 2008.

[48] C. E. Rasmussen and C. K. I. Williams. *Gaussian processes for machine learning.* The MIT Press, 1st. edition, 2006.

[49] Ulrich Reiter, Kjell Brunnström, Katrien De Moor, Mohamed-Chaker Larabi, Manuela Pereira, Antonio Pinheiro, Junyong You, and Andrej Zgank. Factors Influencing Quality of Experience. In *Quality of Experience*, pages 55–74. Springer International Publishing, 2014.

[50] Haakon Riiser, Tore Endestad, Paul Vigmostad, Carsten Griwodz, and Pal Halvorsen. Video Streaming Using a Location-Based Bandwidth-Lookup Service for Bitrate Planning. *ACM Transactions on Multimedia Computing, Communications, and Applications*, 8(3):1–19, 2012.

[51] Michael Seufert, Sebastian Egger, Martin Slanina, Thomas Zinner, Tobias Hossfeld, and Phuoc Tran-Gia. A Survey on Quality of Experience of HTTP Adaptive Streaming. *IEEE Communications Surveys & Tutorials*, 17(1):469–492, 2014.

[52] Kamal Deep Singh, Yassine Hadjadj-Aoul, and Gerardo Rubino. Quality of Experience Estimation for Adaptive HTTP/TCP Video Streaming Using H.264/AVC. In *Proc. of IEEE Consumer Communications and Networking Conference (CCNC)*, pages 127–131, Las Vegas, NV, USA, 2012.

[53] Iraj Sodagar. The MPEG-DASH Standard for Multimedia Streaming Over the Internet. *IEEE Multimedia*, 18(4):62–67, 2011.

[54] Wei Song and Dian W. Tjondronegoro. Acceptability-Based QoE Models for Mobile Video. *IEEE Transactions on Multimedia*, 16(3):738–750, 2014.

[55] Thomas Stockhammer. Dynamic Adaptive Streaming over HTTP – Standards and Design Principles. In *Proc. of ACM Multimedia Systems Conference (MMSys)*, pages 133–144, San Jose, CA, USA, 2011.

[56] Gary J. Sullivan and Thomas Wiegand. Rate-Distortion Optimization for Video Compression. *IEEE Signal Processing Magazine*, 15(6):74–90, 1998.

[57] Viswanathan Swaminathan. Are We in the Middle of a Video Streaming Revolution? *ACM Transactions on Multimedia Computing, Communications, and Applications*, 9(1):1–6, 2013.

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

[59] Truong Cong Thang, Hung T. Le, Anh T. Pham, and Yong Man Ro. An Evaluation of Bitrate Adaptation Methods for HTTP Live Streaming. *IEEE J. on Selected Areas in Communications*, 32(4):693–705, 2014.

[60] Guibin Tian and Yong Liu. Towards Agile and Smooth Video Adaptation in Dynamic HTTP Streaming. In *Proc. of ACM Conference on Emerging Networking Experiments and Technologies (CoNEXT)*, pages 109–120, Nice, France, 2012.

[61] Sheng Wei and Viswanathan Swaminathan. Low Latency Live Video Streaming over HTTP 2.0. In *Proc. of Network and Operating System Support on Digital Audio and Video Workshop (NOSSDAV)*, pages 1–6, New York, NY, USA, 2014.

[62] Thomas Wiegand, Gary J. Sullivan, Gisle Bjontegaard, and Ajay Luthra. Overview of the H. 264/AVC Video Coding Standard. *IEEE Trans. Circuits and Systems for Video Technology*, 13(7):560–576, 2003.

[63] Xiaoqi Yin, Vyas Sekar, and Bruno Sinopoli. Toward a Principled Framework to Design Dynamic Adaptive Streaming Algorithms over HTTP. In *Proc. of 13th ACM Workshop on Hot Topics in Networks (HotNets)*, pages 1–7, Los Angeles, CA, 2014.

[64] Liu Yitong, Shen Yun, Mao Yinian, Liu Jing, Lin Qi, and Yang Dacheng. A Study on Quality of Experience for Adaptive Streaming Service. In *Proc. of IEEE International Conference on Communications Workshops (ICC)*, pages 682–686, 2013.

[65] Chao Zhou, Chia Wen Lin, Xinggong Zhang, and Zongming Guo. 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, 2014.

[66] Chao Zhou, Xinggong Zhang, Longshe Huo, and Zongming Guo. A Control-Theoretic Approach to Rate Adaptation for Dynamic HTTP Streaming. In *Proc. of IEEE Visual Communications and Image Processing (VCIP)*, pages 1–6, San Diego, CA, 2012.

[67] Xiaoqing Zhu, Zhi Li, Rong Pan, Joshua Gahm, and Rao Ru. Fixing Multi-Client Oscillations in HTTP-based Adaptive Streaming: A Control Theoretic Approach. In *Proc. IEEE 15th International Workshop on Multimedia Signal Processing (MMSP)*, pages 230–235, Pula, Italy, 2013.

[68] Xuan Kelvin Zou, Jeffrey Erman, Vijay Gopalakrishnan, Emir Halepovic, Rittwik Jana, Xin Jin, Jennifer Rexford, and Rakesh K Sinha. Can Accurate Predictions Improve Video Streaming in Cellular Networks? In *Proc. of the 16th International Workshop on Mobile Computing Systems and Applications (HotMobile)*, pages 57–62, Santa Fe, NM, 2015.

Appendix A.

Information Contained in the Traces

Each of the collected traces contains several types of information. First, it contains the first 96 bytes of each incoming TCP packet belonging to the monitored TCP flow. Second, it contains the first 512 bytes of each received IEEE 802.11 frame independent of its destination address, including radiotap headers[1] that contain internal MAC information, such as the retransmission flag, the Modulation and Coding Scheme (MCS), and the Signal Strength Indicator (SSI). Except for the radiotap headers, the captured frames are encrypted. Finally, the traces contain periodically recorded values of internal TCP variables, obtained using the tcp_info data structure via the socket interface. From the traces, we computed time series containing various statistics from overlapping time intervals of 1 s to 10 s duration shifted with a step size of 1 s. In addition to throughput statistics, we computed statistics of cross-layer information such as for TCP: delay jitter statistics and the statistics of outstanding bytes, and, for MAC: the number of own frames received, the number of other frames received, MCS and SSI statistics, and the statistics of retransmissions. Figure A shows the throughput, averaged over one second intervals, of one complete trace with a CV of 0.879 recorded at a busy outdoor hotspot of a major German telecommunications operator.

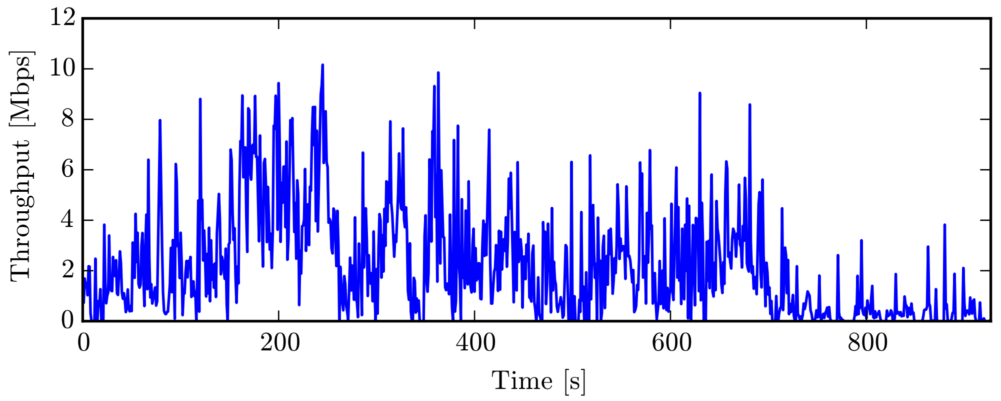

Figure A.1.: A complete example trace with a CV of 0.879, recorded at a busy outdoor hotspot of a major German telecommunications operator.

[1]http://www.radiotap.org

Appendix B. Prediction Methods

We abbreviate the methods by ⟨type⟩:⟨n⟩:⟨parameters⟩, where ⟨type⟩ is the name of the method, n is the number of past throughput values used as input, and ⟨parameters⟩ include further optional configuration parameters.

B.1. Simple moving average

SMA is one of the simplest prediction methods. The predicted value is the average over a number of past measurements. The configuration parameters are the number of past measurements and the type of the used mean value: arithmetic, geometric, or harmonic. In the following, we abbreviate this method with SMA:⟨n⟩:⟨mean type⟩, where $n \geq 1$ is the number of past measurements, and 'mean type' is one of $\{\mathrm{ar}, \mathrm{gm}, \mathrm{hm}\}$. For example, SMA:2:ar means that the predicted value is the arithmetic mean from two past measurements. In particular, we denote the naïve approach of using the most recent measurement as the predicted value with SMA:1:ar.

B.2. Linear extrapolation

Linear extrapolation is another straightforward prediction method that differs from SMA in that it takes into account the linear trend from the past measurements. More specifically, linear extrapolation fits a linear curve into the set of given past measurements, minimizing the MSE, and computes the prediction from extrapolating the curve to the prediction horizon. It thus requires at least two past measurements to compute a prediction. We abbreviate linear extrapolation with LinExt:⟨n⟩, where $n \geq 2$ is the number of past measurements.

B.3. Double exponential smoothing

Similar to linear extrapolation, double exponential smoothing tries to account for the trend in the data. In the following, we use a variant of the method, usually referred to as Holt-Winters double exponential smoothing. With Holt-Winters, for the given past measurements $x_1, \ldots, x_n$, the prediction is computed as $x_{n+1} = a_n + b_n$, where a_n, b_n are computed by the following recursive procedure.

$$\begin{aligned} a_n &= \alpha x_n + (1-\alpha)(a_{n-1} + b_{n-1})\,, \text{ for } n > 2 \\ b_n &= \beta(a_n - a_{n-1}) + (1-\beta)b_{n-1}\,, \text{ for } n > 2\,, \end{aligned}$$

with $a_2 = x_2$, and $b_2 = x_2 - x_1$.

The Holt-Winters method has configuration parameters α and β that strongly influence the prediction quality and thus have to be carefully tuned. In our work, we tune them for each prediction by minimizing the MSE within the past measurements, which is given by $\frac{1}{n-2}\sum_{k=3}^{n}(x_k - (a_{k-1} + b_{k-1}))^2$. Thus, this method requires at least three past values to compute a prediction. As abbreviation, we use HW:$\langle n \rangle$:mse, where $n \geq 3$ is the number of the last values, and *mse* indicates the approach used to tune α and β.

Appendix C.

Correlation of Underestimations and Overestimations

As described in Section 6.3, we observed that in all the studied traces the probability for occurrences of underestimations and overestimations exhibit significant temporal correlation. In particular, the probability that an underestimation is followed by an overestimation and vice versa is significantly over 50% for most traces for all time scales, exceeding 80% or even 90% in some cases. This observation is directly related to the distinct negative correlation of the throughput process after differencing, as depicted in Figure 5.1. The distribution of per-trace values is depicted in Figure C.1.

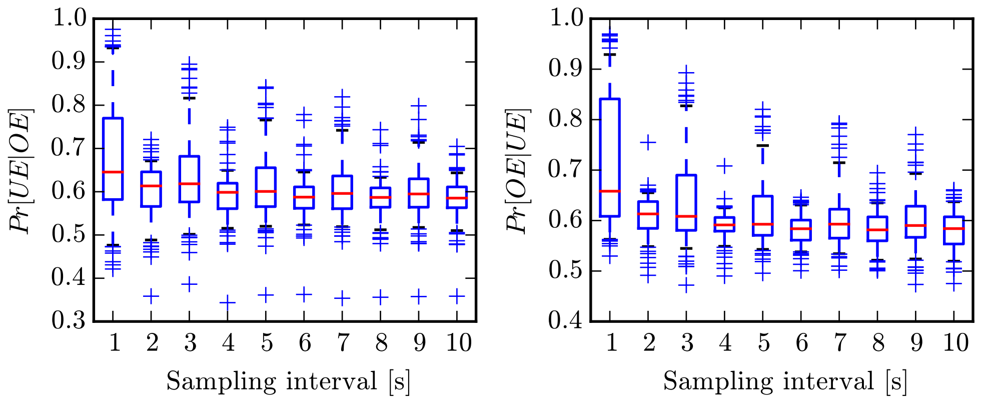

Figure C.1.: Per-trace probability that an underestimation is followed by an overestimation and vice versa. Horizontal line: median, box: quartiles, whiskers: 0.5 and 0.95 quantiles, flier points: outliers. See Section 6.3 for details.

Appendix D.

Fitting Prediction Error Distributions

For the underestimation errors, distributions are truncated to the range $[0, 1]$, and for the overestimation errors, to the range $[0, \infty)$. The CDF $F_{\text{tr}}(\cdot)$ of a distribution truncated to $[a,\, b]$ is obtained from the original CDF $F(\cdot)$ as $F_{\text{tr}}(x) = \frac{F(x)-F(a)}{F(b)-F(a)},\ \ x \in [a,\, b]$.

We fit a distribution to the data by minimizing the squared distance (L^2-norm) between its CDF and the truncated ECDF. In order to make the fit more precise in the range which is relevant for adaptive streaming clients, we truncate ECDF's to the interval $[0.1, 5.0]$ for the overestimation errors, and to the interval $[0.1, 1.0]$ for the underestimation errors. Afterwards, Kolmogorov-Smirnov test is used to verify the goodness of the fit [19].

The results are shown in Figure D.1. The CDF's are fitted to ECDF's over the joined set of data points from all traces. It turns out that both the underestimation and the overestimation errors are extremely well represented by a Lomax distribution. These findings are consistent with those obtained by fitting the prediction errors from individual traces, which are omitted here.

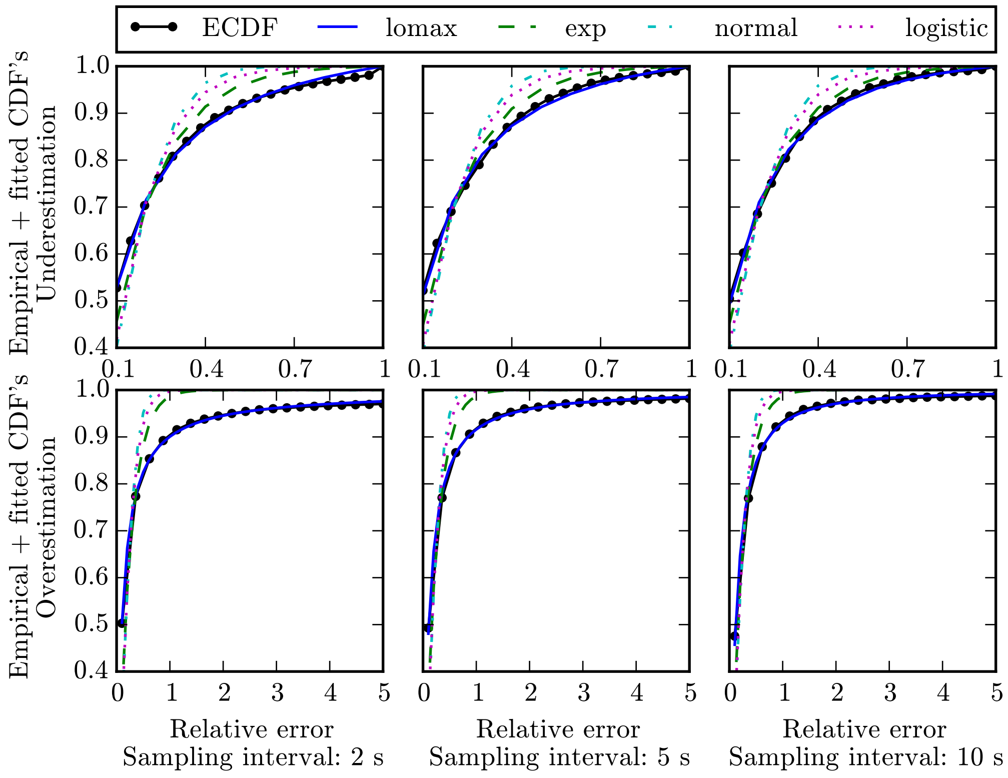

Figure D.1.: Fitting distributions for relative prediction errors. See Section 6.4 for details.