Hindawi Publishing Corporation
Advances in Multimedia
Volume 2014, Article ID 805852, 21 pages
http://dx.doi.org/10.1155/2014/805852

Research Article

Adaptive Media Streaming to Mobile Devices: Challenges, Enhancements, and Recommendations

Kristian Evensen,[1,2] Tomas Kupka,[3] Haakon Riiser,[4] Pengpeng Ni,[1,5] Ragnhild Eg,[1,6] Carsten Griwodz,[1,5] and Pål Halvorsen[1,5]

[1] *Simula Research Laboratory, P.O. Box 134, 1325 Lysaker, Norway*
[2] *Celerway Communications, Martin Lingesvei 17, 1364 Fornebu, Norway*
[3] *Telenor Digital (Comoyo), P.O. Box 800, 1331 Fornebu, Norway*
[4] *Opera Software, Gjerdrums Vei 19, 0484 Oslo, Norway*
[5] *Department of Informatics, University of Oslo, P.O. Box 1080, Blindern, 0316 Oslo, Norway*
[6] *Department of Psychology, University of Oslo, P.O Box 1094, Blindern, 0317 Oslo, Norway*

Correspondence should be addressed to Kristian Evensen; kristrev@simula.no

Received 12 March 2014; Revised 28 June 2014; Accepted 28 June 2014; Published 10 September 2014

Academic Editor: Marco Roccetti

Video streaming is predicted to become the dominating traffic in mobile broadband networks. At the same time, adaptive HTTP streaming is developing into the preferred way of streaming media over the Internet. In this paper, we evaluate how different components of a streaming system can be optimized when serving content to mobile devices in particular. We first analyze the media traffic from a Norwegian network and media provider. Based on our findings, we outline benefits and challenges for HTTP streaming, on the sender and the receiver side, and we investigate how HTTP-based streaming affects server performance. Furthermore, we discuss various aspects of efficient coding of the video segments from both performance and user perception point of view. The final part of the paper studies efficient adaptation and delivery to mobile devices over wireless networks. We experimentally evaluate and improve adaptation strategies, multilink solutions, and bandwidth prediction techniques. Based on the results from our evaluations, we make recommendations for how an adaptive streaming system should handle mobile devices. Small changes, or simple awareness of how users perceive quality, can often have large effects.

1. Introduction

Smartphones and tablets have developed into popular devices for streaming media. For example, YouTube [1] reports that their traffic from mobile devices tripled in 2011 and that more than 20% of the global YouTube views took place on mobile devices. Cisco's Visual Networking Index [2] ranks video traffic to be the fastest growing traffic type in mobile broadband networks, with a predicted 16-fold increase in mobile video streaming between 2012 and 2017. Such an increase would imply that video streams will make up two-thirds of the world's mobile data traffic by 2017.

The main idea of adaptive streaming over HTTP is to deliver video by splitting the original stream into independent segments of a specified length. The segments are encoded in multiple qualities (bitrates) and uploaded to web servers. Segments are downloaded like traditional web objects, and a client can select bitrates for individual segments based on, for example, observed resource availability. Adaptive HTTP streaming has many advantages compared to traditional streaming techniques, for example, NAT-friendliness and TCP's congestion avoidance, as well as the existing infrastructure's scalability using caches and content distribution networks. Furthermore, adaptive HTTP streaming is supported by major industry actors and has been implemented in systems such as Microsoft's Smooth Streaming [3], Adobe's HTTP Dynamic Streaming [4], and Apple's HTTP Live Streaming [5]. This kind of streaming is also ratified for an international standard by ISO/IEC, known as MPEG Dynamic Adaptive Streaming over HTTP

(MPEG-DASH) [6]. The technology has been used to stream major events like the 2010 Winter Olympics [3], the 2010 FIFA World Cup [7], and the NFL Super Bowl [7] to millions of concurrent users. Adaptive HTTP streaming is also used by popular streaming services such as Netflix, HBO, Hulu, Viaplay, TV2 Sumo, and Comoyo.

Despite the recent popularity, large-scale HTTP streaming solutions are relatively new, and there are many challenges left to solve. For example, segmented video streaming causes a distinct traffic pattern (on/off) different from most HTTP-based traffic, and TCP's unicast-nature is not a good match for the limited resources in mobile broadband networks. Moreover, mobile devices are heterogeneous, which means that one quality scheduling algorithm will not provide the optimal experience for all devices. Also, the bandwidth in wireless networks, especially mobile broadband networks, fluctuates more than in fixed networks [8, 9]. In this paper, we address some of the key challenges in such a streaming scenario with a main focus on mobile devices. In particular, we discuss performance related aspects of the entire streaming pipeline, from the sender machine to the receiving device. We have evaluated the different components by conducting a series of simulations and real-world experiments. From the experimental results, we put forward suggestions for a range of enhancements to improve streaming performance. As a worst-case scenario with high workload peaks and a large number of concurrent users, Section 3 presents an analysis of user statistics from a large live streaming event. These statistics demonstrate the benefits of HTTP streaming from a service provider's perspective. We then look at how HTTP-based streaming affects server performance in Section 4, while Section 5 focuses on video coding and adaptation with respect to server performance and user perception. Section 6 provides an analysis of streaming performance and quality (bitrate) adaptation schemes, whereas Section 7 outlines how HTTP streaming can benefit from bandwidth aggregation. Section 8 presents ideas and suggestions on how to improve streaming quality by using bandwidth lookup services. Finally, Section 10 summarizes and concludes the work by highlighting the core ideas.

2. Related Work

2.1. HTTP Adaptive Streaming. Video streaming is a bandwidth intensive service, which typically requires that providers make large investments in infrastructure. With cost-effective solutions that reuse existing infrastructure, HTTP has become the de facto protocol for adaptive streaming of video content, and adaptive HTTP streaming is now widely deployed by major systems provided by, for example, Microsoft [3], Adobe [4], and Apple [5].

With HTTP adaptive streaming, media players are able to download a segment in a quality (bitrate) that matches resource availability both in the network and on end systems. Consequently, the player can trade off quality for a more robust playout. For example, if the media player selects a video quality where the bitrate is lower than the current download rate, the unused bandwidth may serve to fill the buffers and avoid playout stalls. An adaptation strategy that aims at the right trade-off must take a multitude of factors into account. These include average quality, frequency of quality switches, maximum buffer size, and prediction of the rate of download for the following segment.

Adaptation strategies for this kind of segmented HTTP streaming have recently become a hot research topic. For a wired network scenario, there are several studies on the effectiveness of rate-adaptation algorithms in the existing systems of Microsoft, Adobe, and Apple and a variety of less prominent systems (e.g., [9, 12, 13]).

Concurrently, with these studies of the state of the art, the search for algorithms that ensure long-term stable quality has commenced. Researchers have held the view that long-term stability is beneficial for users' quality of experience. A study that has supported this belief was conducted by Zink et al. [14]. With adaptive HTTP streaming in mind, Tavakoli et al. [15] conducted a new study on this subject and found that quality increase yields lower QoE than constant quality at high bandwidth, while this is not necessarily the case for constant quality at low bandwidth. Decreasing quality, however, was found to be generally disruptive to QoE. Borowiak and Reiter [16] found indications that high activity in the content decreases quality requirements over time. In spite of this, we believe that the rule of thumb that targets long-term constant quality can still be considered valid.

This goal has been pursued by Akhshabi et al. [17], who developed the client-side AdapTech mechanism to address the various problems of the 3 main commercial HTTP adaptive streaming systems. This was also the goal of Jiang et al. [18] and our own client-side reactive algorithm [10]. Akhshabi et al. [19] proposed also a server-side traffic shaping to stabilize the oscillation of streams. Miller et al. [20] aim at a trade-off between startup latency and avoiding quality switches, while Li et al. [21] observed that streaming with Microsoft Smooth Streaming led to synchronized quality swings for multiple streams sharing a bottleneck and developed PANDA to compensate for this. Houdaille and Gouache [22] apply traffic shaping with the goal of achieving stable quality.

2.2. Wireless HTTP Adaptive Streaming. Pu et al. [23] propose a proxy that can perform adaptation between wired and wireless networks to increase fairness for HTTP adaptive streaming to wireless clients. The importance of this work is demonstrated by Mansy et al. [24], who evaluated mobile HTTP adaptive streaming to various mobile phone operating systems. They observed basic differences in delivering of the same service to different platforms and demonstrated that they lead to unfairness. Furthermore, Siekkinen et al. [25] showed results that imply that the bursty nature of HTTP adaptive streaming can be used for the benefit of power consumption in wireless network. While Pu et al. [23] use a proxy server, Havey et al. [26] created a receiver-driven rate-adaptive algorithm for wireless streaming. Also our work avoids middleboxes for streaming smoothly to mobile clients [27]. Rebuffering as the single most inhibitive factor to QoE motivated the approach by Oyman and Singh [28].

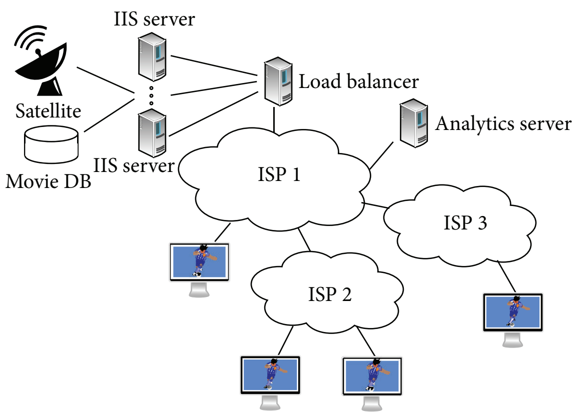


FIGURE 1: The Comoyo streaming infrastructure.

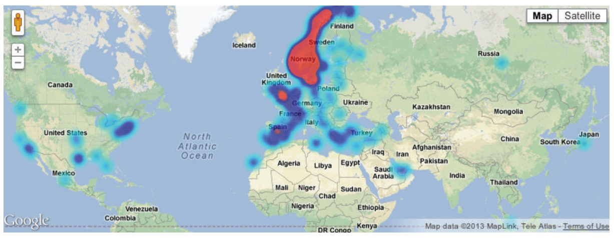


FIGURE 2: Heat-map of the geographical IP distribution in the world (the highest density of clients is in the red areas). There are also some clients in Russia and Japan outside the shown map.

3. HTTP Streaming: A Providers' Perspective

To investigate the potential benefits of HTTP streaming for service providers, we evaluated the behavior of the Smooth Streaming [3] system used by the Norwegian TV/movie provider Comoyo. Here, we present results from an analysis of log files from 8 live-streamed Norwegian premier league soccer matches (more details can be found in [29]). The client- and server-side logs were collected on May 23, 2012, but we have made similar observations on other dates and in on-demand scenarios.

The Comoyo network infrastructure, illustrated in Figure 1, is a typical HTTP streaming system for on-demand and live services. Microsoft's IIS media server is run on several machines, which are placed behind a load balancer. Incoming requests are then distributed across servers, according to a proprietary scheduler.

The connections to the Comoyo servers originated from 194 different network providers, and we logged 6567 unique client IP addresses. As expected, the majority of clients were located in Norway, as depicted in Figure 2. Nevertheless, users were located worldwide, distributed across 562 cities in 36 countries.

In traditional streaming systems, there has been a one-to-one ratio between the number of active users and users that were receiving content directly from the server. Our analysis of the Comoyo log files shows that this is not the case with HTTP streaming. Compared to the 6567 unique client IP addresses we observed, in the client-side logs, only 1328 were logged at the media servers. Hence, a large amount of the traffic is handled by existing infrastructure like proxy caches. This observation is strengthened by the fact that the media servers provided roughly 22% of the estimated number of bytes received by clients. In other words, HTTP streaming reduces the need for providers to make large infrastructure investments.

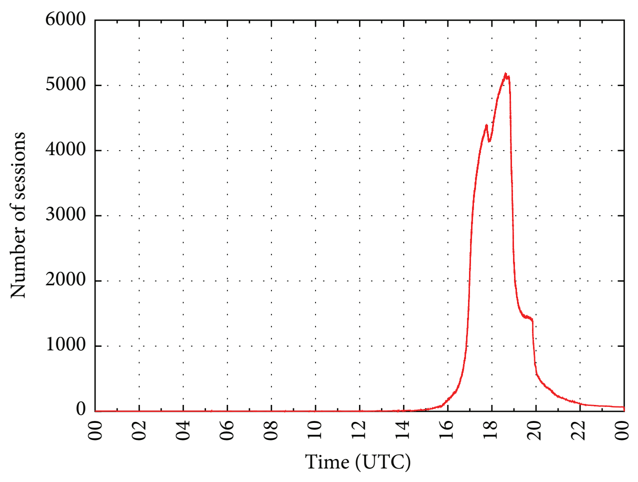
An analysis of the client access patterns (Figure 3) revealed that the vast majority of streams commenced when the first match started at 17.00 UTC (Figure 3(a)). As a result, a large number of clients wished to access segments at the same time. Even though the arrival times of viewers are distributed around the start time of the game, this graph does not show whether clients follow the streams live or whether late viewers chose to watch games from their start.

This information can be derived from Figure 3(b), which shows how much time has passed between availability of a segment on the server and the time when it was requested by users of the system. Figure 3(b) shows the CDF only for the 5 most popular games, each represented by one line, but the behavior is representative for all games. The figure shows that about 90% of requests for a particular segment were served within a 10-second period after that segment became available; we can therefore conclude that most viewers chose to follow the games live.

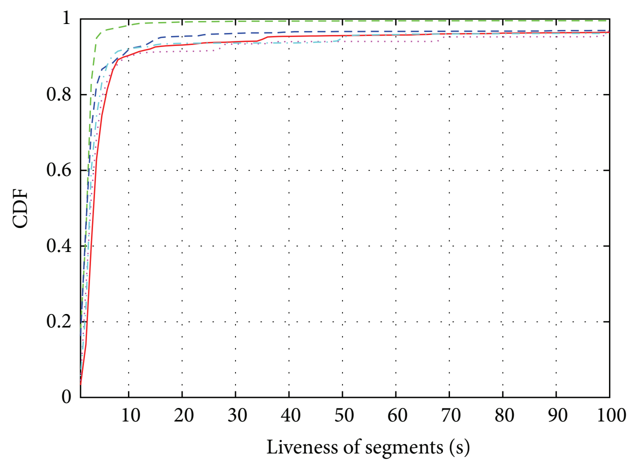
The decision to follow a live stream implies that the majority of viewers try to access a very small number of identical segments through concurrent TCP connections. This is bound to lead to a concentrated on/off workload at the server. While we have observed this behavior for scenario of live football streaming, it has also been reported for new movie releases when release dates have been advertised [30]. Furthermore, Li et al. [21] noticed that streaming sessions can synchronize implicitly even if they have been started at arbitrary times. Accordingly, the next section goes into further details on optimizing the management of concurrent segment requests.

The client logs revealed that about 99.5% of the clients experienced quality (bitrate) switching during their streaming session. As shown in Figure 4(a), the number of bitrate switches during a session varied from a couple to well over 100. Furthermore, more than half of the sessions experienced at least one buffer underrun with a related playout stall (Figure 4(b)). Underruns such as these are in most cases due to inefficient video adaptation. Adaptation algorithms might, for example, not be designed to properly consider bandwidth fluctuations. Varying network conditions are a common phenomenon and especially in wireless mobile broadband networks.

In summary, our analysis of the Comoyo system shows the efficiency of adaptive segment streaming. However, some areas still require improvement. On the sender side, better solutions are needed for the management of concurrent segment requests, while, on the receiver side, the number of quality switches and buffer underruns should be reduced. The latter challenges are especially important in mobile scenarios where network availability varies far more than for fixed networks.

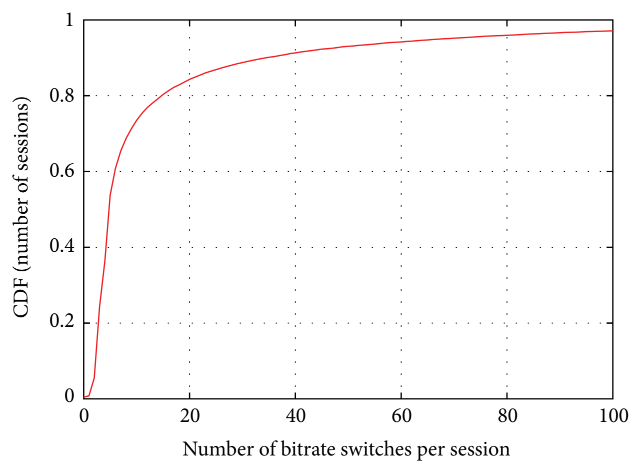


(a) Number of active sessions for all games at each point in time (using a one-minute resolution)

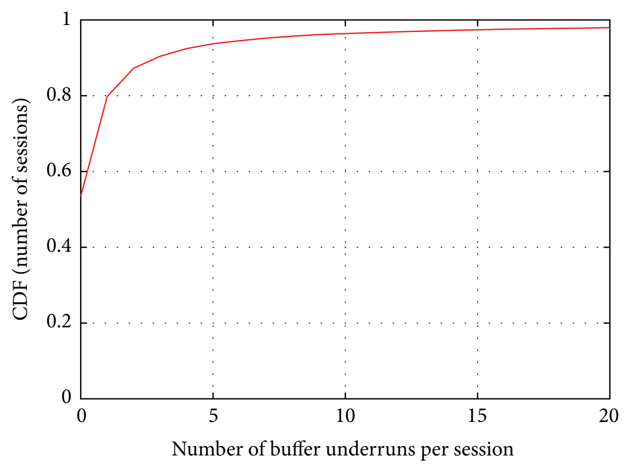


(b) Cumulative distribution function showing the liveness of segments. Each line represents the liveness for one of the 5 most popular games according to the server log

FIGURE 3: Client access patterns.



(a) Number of bitrate switches



(b) Number of buffer underruns

FIGURE 4: Session statistics based on client logs.

4. Concurrent Management of Connections

Performance of the server or the sending machine, either an origin server or a proxy cache, is important to the overall quality of the streaming service. In this respect, there are differences between connections to mobile wireless devices and a machine connected to the wired network. For instance, mobile providers make heavy use of middleboxes to distribute more fairly the limited radio resources amongst the users. Also, smartphone vendors typically set the TCP receiving buffer size to a small value, to compensate for buffer bloat introduced by the middleboxes [31]. However, each device (mobile or middlebox) will speak normal TCP, and even though the actual transmission might differ, the request phase will be the same as for a fixed connection. Thus, at the sending side, it often does not matter whether the client is mobile or not; the machine serves each request equally.

As an example, consider a live event that is streamed to a massive crowd equipped with different types of devices and connected to different types of networks. In this scenario, the servers experience a massive load. Such a scenario relates to our observations from Figure 3(b), where the same segment is served many times over within a very short period of time. In the live scenario, this happens because all clients want to be as live as possible and therefore request a segment as soon as it becomes available. In an on-demand scenario, one might observe the same pattern after the buffer is filled, but not necessarily for the same segment. For a single client, it is well known that a segmented download leads to an on/off network traffic pattern. Typically, a client downloads

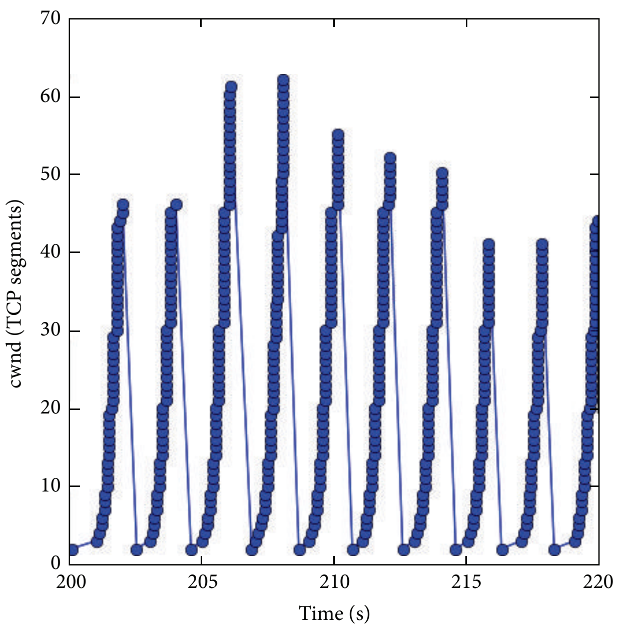


FIGURE 5: Observed TCP congestion window for 2-second segments.

the most live segment and then waits for the next segment to become available. Figure 5 illustrates how the congestion window grows during the on-period and shrinks during the off-period. For the server machines, such accesses might result in a frequent high-load/idle-load pattern.

Several trade-offs and potential performance enhancements originated from these observations. We have evaluated both server-side and client-side modifications. The modifications were evaluated using the well-known ns-2 network simulator, because we could not deploy them in a real-world experiment in the running system of any of our partners. Thus, we perform the evaluation in a lab environment with limited resources. The setup is similar to a one-server version of the infrastructure in Figure 1, where clients access the server over a 100 Mbps bottleneck link (we have found similar trends when testing with a 1 Gbps bottleneck link, with only the total number of clients scaled up; see [32] for details). The delays (RTTs) between the clients and the server are normally distributed with an average of 55 ms and a variance of 5 ms; these values correspond well to those observed for ADSL access networks [33]. The router queue follows the rule of thumb of setting the size to one bandwidth delay product (BDP) and is configured as a drop tail queue, which is one of the most commonly used queuing strategies. Furthermore, we modeled client arrivals as a Poisson process with an average interarrival time: λ = number of clients/segment duration. This means that all clients join the stream within the first segment duration (the segment duration is fixed to 2 seconds; see Section 5.1). This models the case when people are waiting in front of a "Your stream starts in . . ." screen for the start of stream so that they do not miss the beginning. To evaluate the performance of the server, we used liveness and packet loss as main metrics. Liveness measures the duration in time that the client lags behind the live stream (in terms of display latency, after a segment is made available on the server).

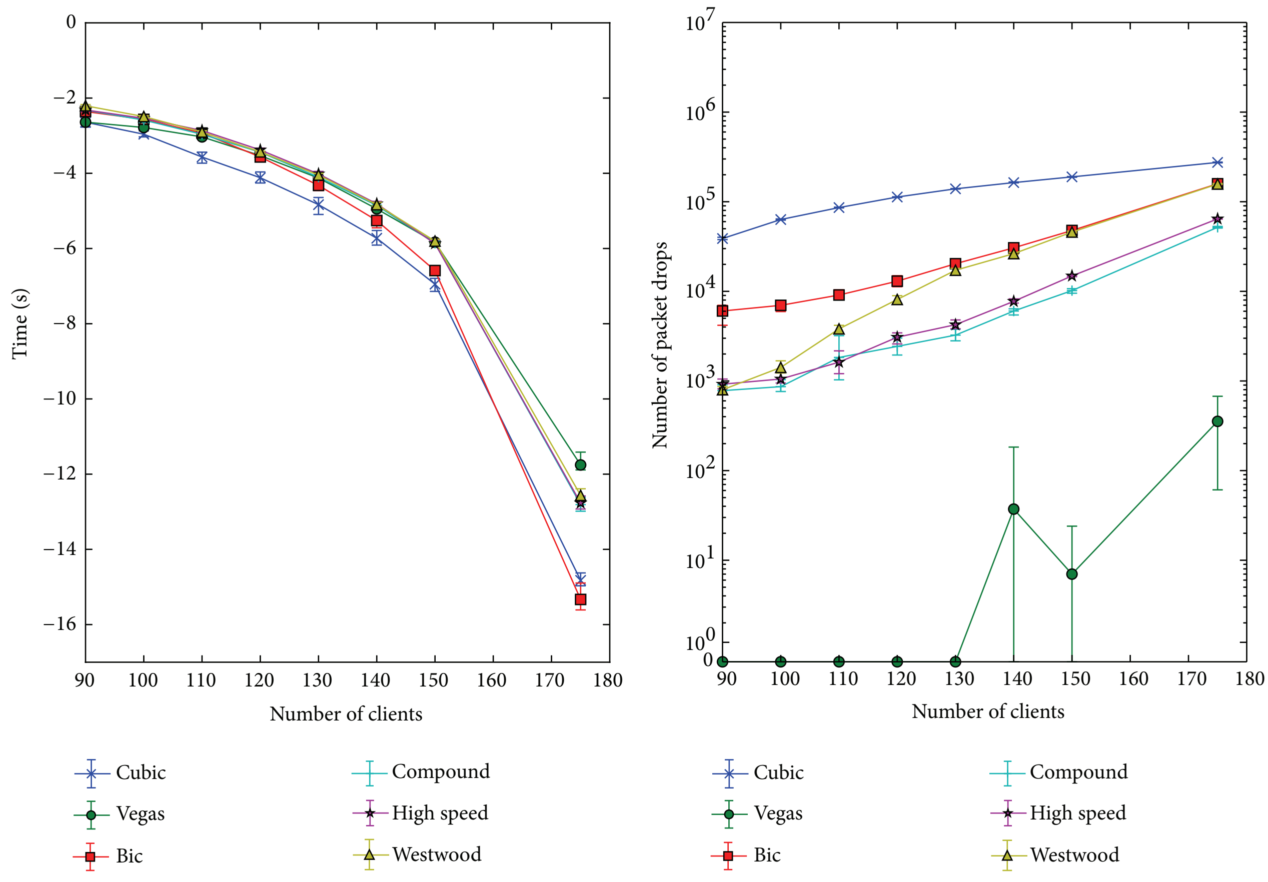
In Figures 6, 7, and 8, the liveness is shown as a snapshot of the client at the end of a 20-minute stream, and it includes initial startup latency and potential stalls. Every experiment was run 10 times with slightly different client interarrival times. The plots show the average value with the error bars as the minimum and maximum values (which in most cases are too small to be seen).

4.1. Performance of TCP Congestion Control. Running on top of TCP, the performance of HTTP streaming is heavily influenced by the TCP congestion control algorithm. In this section, we therefore evaluate how the most common versions cope with the on/off traffic pattern.

Figure 6 shows the achieved average liveness and the number of packet drops across all clients for the evaluated congestion control algorithms. Although the server bandwidth of 100 Mbps should provide enough bandwidth for smooth playout for around 200 clients, the liveness graph shows that this is not the case. As the clients number grows, the liveness decreases due to multiple playout stalls. The reason for this inefficiency is found in the overflowing of the router queue. When the queue is full, incoming data packets are discarded (Figure 6(b)) and must be retransmitted. We also observe that the more aggressive, loss-based variants of TCP congestion control algorithms, like Cubic and Bic, generate the highest losses and have the worst liveness. This is due to higher competition for the resources during the on-periods, resulting in higher loss rates. An interesting congestion control alternative is Vegas, which backs off before the bottleneck queue overruns. We see that Vegas also performs better than Cubic in terms of liveness and can cope with an increased number of clients better. However, Vegas has been shown [34] to perform badly when competing with loss-based congestion controls. Therefore, unless the majority of traffic through the bottleneck consists of TCP connections using Vegas, one must consider the deployment of Vegas very carefully. In the remaining experiments, we therefore use default Linux congestion control (Cubic) [35].

4.2. Requests Distributed over Time. In Section 3, we observed that a competition for resources occurs at the server because clients often download a segment as soon as it becomes available. This type of segment request synchronization leads to reduced performance since many clients hit the on-periods at the same time, while the off-periods leave the machine idle.

To avoid this synchronization, we propose to distribute the requests over one segment duration. There are several ways to achieve this, but we aim for no additional load on the server. With our approach, the clients check the media presentation description for the most recent segment following the start of a session. After that, a new segment is assumed to be produced for every segment duration. When the segment duration has passed, the next segment is requested. Since the initial time for availability of a segment differs between clients, the requests stay distributed over time.



(a) Liveness

(b) Number of dropped packets

Figure 6: Performance of alternative TCP congestion control algorithms.

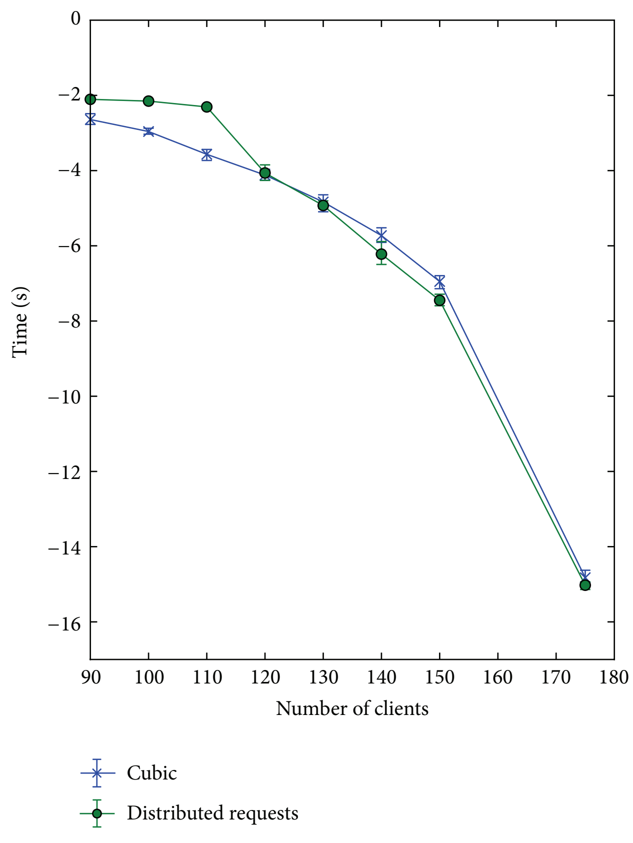
In our experiment, the requests are exponentially distributed over the entire segment duration. The results show that distributed requests increase the liveness when the number of clients is small, while it remains largely unchanged with a larger number of clients (Figure 7(a)). However, the number of packet losses is lower for distributed requests (Figure 7(b)), providing a better utilization of network resources.

4.3. Limited Congestion Window. Both live and on-demand scenarios display similar on/off patterns, and, in this case, a fast download of a segment prolongs the wait period. Hence, there is no need for the client to download a segment as quickly as possible, as long as it is downloaded in time for playout. Furthermore, TCP's bandwidth sharing is fair for long running data transfers. However, for short transfers, the sharing can become unfair. To reduce this unfairness, we have explored the effects of limiting the server-side TCP congestion window. The limited congestion window can lead to longer download times, thereby reducing off-periods and resulting in a behavior similar to TCP pacing [36] or server-based traffic shaping [19]. To avoid playout stalls due to congestion window limitation, we chose a congestion window that would allow for a segment to be easily downloaded in one segment duration [32]. The congestion window limit was set to 20 TCP segments, which equals a bandwidth 3 times larger than the average bitrate of the stream (to account for bitrate variance). Figure 8(a) shows that this approach improves the average liveness. Furthermore, from Figure 8(b), we observe a significant reduction in the number of dropped packets. This reduction also indicates a lighter load on the bottleneck router, resulting in a more efficient resource utilization.

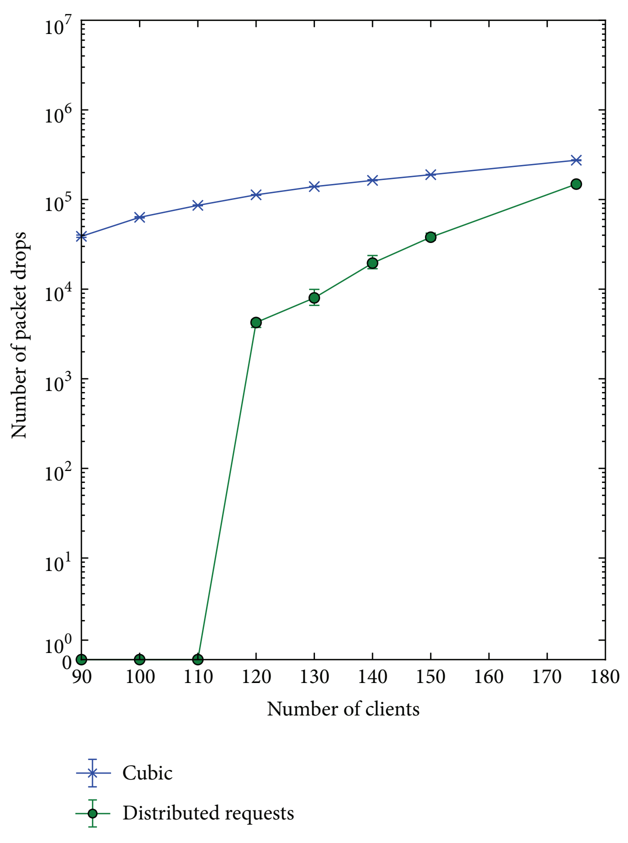
In summary, simple changes to server parameters like TCP congestion control and the client-side request strategy can lead to increased performance in terms of both better liveness and video quality (see [32] for more details).

5. Video Coding for Mobile Streaming

The choices with respect to video coding strongly influence the quality of the received video. For example, the length of the segments affects the encoding efficiency and the adaptation points, and the parameters used to code video in different qualities often determine the visual quality of the video. Furthermore, as each segment is wrapped with



(a) Liveness



(b) Number of dropped packets

FIGURE 7: Performance of regular versus distributed requests.

metadata, the size of the container determines the effective payload used for video, that is, again impacting the perceived quality. In this section, we discuss various trade-offs in this context.

5.1. Segment Lengths. Video segment duration influences the performance of media streaming in several ways, from quality adaptation frequency and number of requests and files to handle to coding efficiency and liveness of streams. Different systems use different segment lengths that typically vary from 2 to 10 seconds; for instance, Microsoft uses 2–4 seconds in Smooth Streaming, while Apple recommends segment length of about 10 seconds. The duration of segments has been discussed briefly before [9], but here we give an evaluation of efficiency from the network perspective and the perceived user experience.

From the *network point of view*, we know that the length of a segment is tied up with the efficiency of congestion control, as outlined in Section 4. To explore the effects of segment duration, we used the same setup to run simulations with the industry standard 2- and 10-second segments.

The trace of a congestion window for the 10-second segments is plotted in Figure 9. Compared to the 2-second segment scenario in Figure 5, we see that the window size oscillates with a lower frequency relative to the segment size. Note here that 10-second segments are 5 times longer in duration, but, due to increased compression efficiency, they are not necessarily 5 times larger in size. Nevertheless, longer duration segments are larger than shorter ones and therefore require more time to download. Prolonged download (on) and idle (off) periods provide TCP with more time to reach its operating state. Figure 9 also exhibits the characteristic curve of Cubic [35] when it probes for available bandwidth, which the limited time of the 2-second scenario (Figure 5) does not allow.

Concerning performance, Figure 10 portrays the liveness and packet drops for a live stream. Our experiments show better liveness for the 2-second scenario, in part due to the distribution of client arrivals across one segment duration. Client arrival times are generally higher for the 10-second segments, increasing the average startup times and decreasing the liveness. Both scenarios lose liveness in a similar manner as the number of clients increases.

Figure 10(b) is surprising. Although Figure 9 shows that 10-second segments allow each TCP stream to leave slow start and enter congestion avoidance mode, this is not the case for 2-second segments (Figure 5), and although the queue length is identical to one BDP in both cases, we can see from Figure 10(b) that the 10-second segments lead to a higher packet loss rate for a smaller number of clients. For the case where client requests are distributed

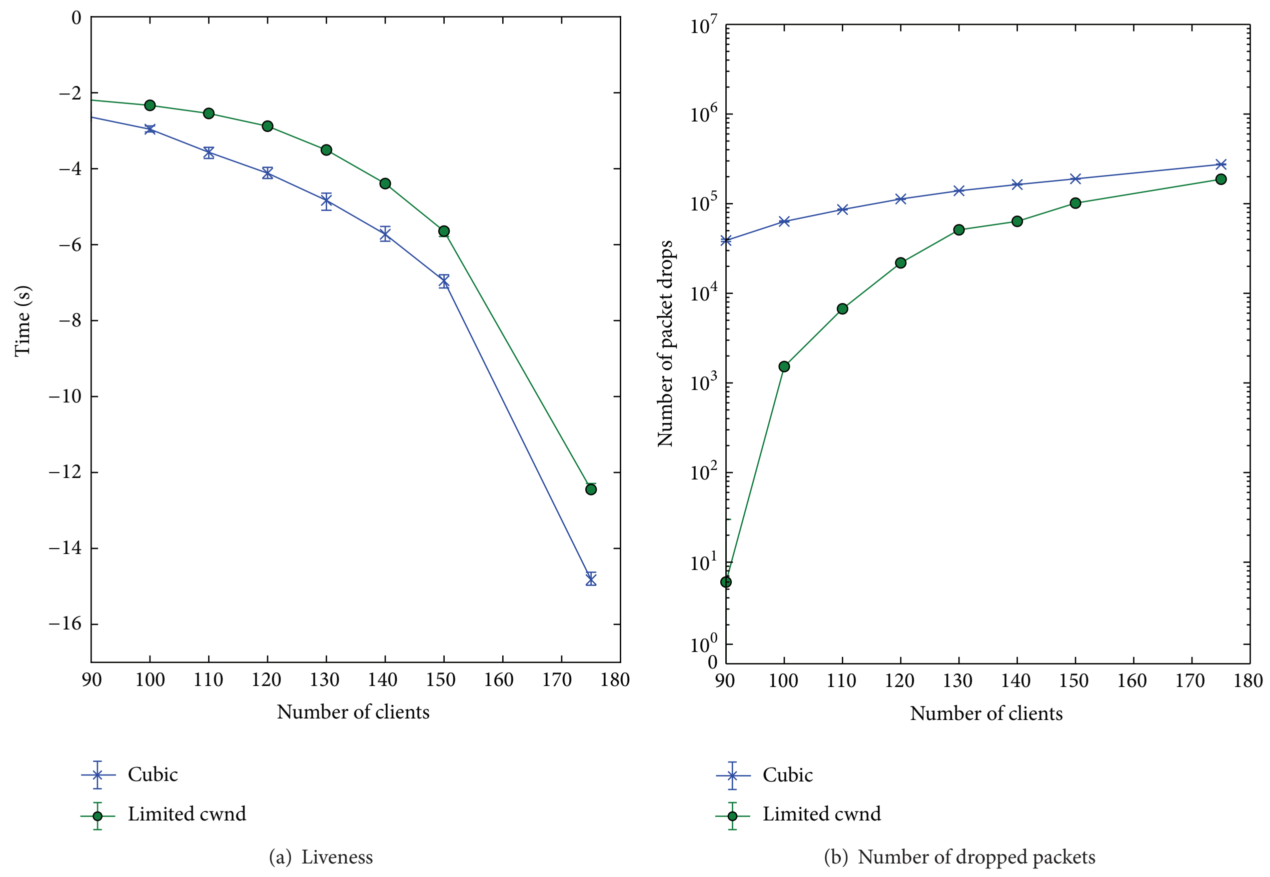


FIGURE 8: Performance of a limited TCP congestion window.

over the entire segment duration, both Esteban et al. [37] and Kupka et al. [32] showed that longer segments lead to higher average quality, which would contradict this finding. However, Figure 3(b) shows that live requests concentrate right after a segment becomes available. The result is more competitions and a queue that is full when sized at the BDP. Consequently, we argue for 2-second segments as the better alternative for the live streaming scenario.

While short segment lengths are beneficial with respect to bandwidth adaptation, our studies on perceived quality of video streams show that there is a perceptual limit to how far segment durations can be reduced. From the *user perspective*, very short segments may introduce rapid quality changes as the stream adapts to the available bandwidth. To study the relation between perceived video quality and frequent segment switches, we ran a series of subjective tests for different quality adaptation techniques. Specifically, we explored the flicker effect [38], an artifact that mimics the visual consequence of frequent bitrate adaptation. A total of 28 assessors took part in the subjective evaluation tests, which were conducted in a mobile scenario. Assessors rated the quality of 12-second-long videos presented on 3.5-inch iPhones with 480 × 320 resolution screens. We used 4 different video sequences, selected to include contents with both high and low motion and spatial detail. The tests included 3 different adaptation techniques: compression, resolution, and frame rate. Video compression was implemented with the H.264 encoder's quantization parameter (QP), using compression rates that ranged from QP12 (best quality) to QP40 (worst quality). The resolution was set to 480 × 320 pixels or downscaled to 240 × 160 or 120 × 80 pixels, and the frame rate was varied from 30 fps to 3 fps. Videos were presented with quality changes occurring at regular intervals, between 0.2 and 3 seconds. Flicker sessions were also compared to sessions with constant high or low quality. Following each video presentation, assessors were first asked whether they perceived the video quality to be stable. Following this step, they were prompted to give an acceptance score for the video quality using the ITU-T P.910 Absolute Category Rating (ACR) method [39]. P.910 ACR defines a 5-point assessment scale with the labels excellent, good, fair, poor, and bad. Video sequences must be randomly ordered for each test subject.

To illustrate our findings, mean acceptance scores from the subjective assessments are plotted in Figure 11. Each subfigure refers to one test series and demonstrates the acceptance of scaling with one adaptation technique. Each series included constant-quality reference videos at both the highest quality (HQ) and lowest quality (LQ) for each content type. The references were part of the randomly ordered series. The x-axis of all subfigures shows the time between quality changes in seconds.

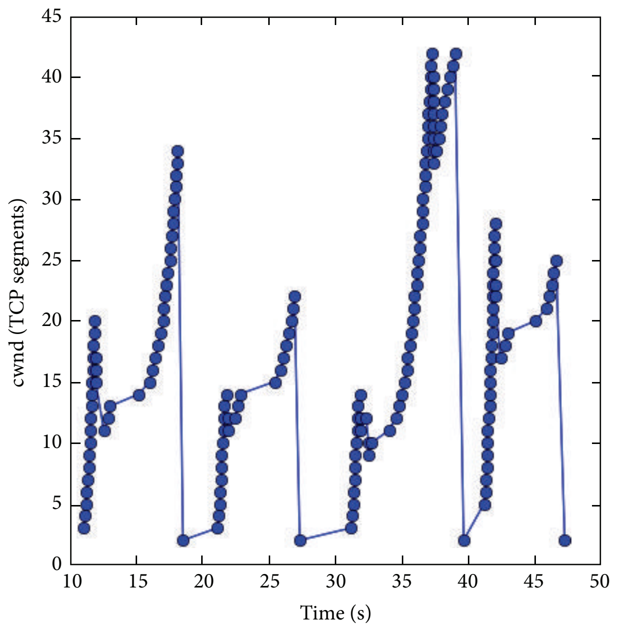


Figure 9: Observed TCP congestion window with 10-second segments.

Figure 11(c) reveals that adaptation in the temporal dimension is considered acceptable for all periods with frame rates at or above 15 fps. On the other hand, the flicker effect is quite pronounced for frequent quality changes in the spatial dimension, as seen in Figure 11(a) for compression and Figure 11(b) for resolution. Nevertheless, the influence of spatial flicker on perceived quality diminishes as the periods increase; that is, the time between quality changes grows. Important in the context of adaptive TCP streaming is the observation that sessions are rated worse than sessions where the quality is kept constantly low when the quality is changed more frequently than once every second. When segment switches occur at intervals that are 1 second or less frequently, mean acceptance scores for many quality shift levels are higher than the constant low quality. At 2 seconds and above, this trend is more or less established, with most quality shifts rated higher than those kept at a stable but low quality level. Figure 11(d) illustrates that this holds true across different content types as well.

Combined, the studies on segment duration from both the network and the user perspective highlight the benefits of 2-second segments. Shorter durations lead to unacceptable quality ratings, poor coding efficiency, and high load in terms of requests. Longer segments increase encoding efficiency and reduce the number of requests but give longer response times for adaptation with the liveness going down. The industry's standard range of 2–10 seconds will, from our experience, yield reasonable network efficiency, yet our studies favor the lower part of the range.

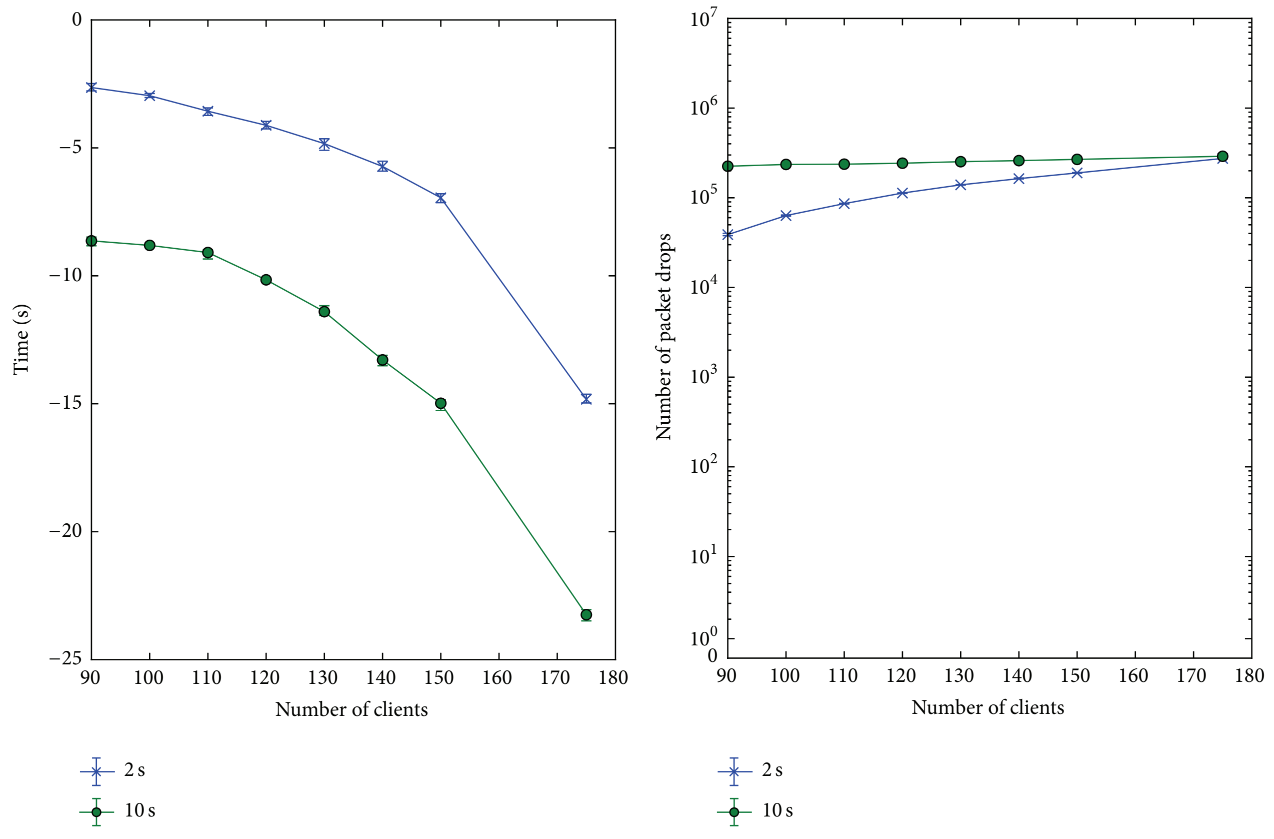
5.2. Perception and Video Adaptation. While the rate of quality adaptations may influence perceived video quality due to the resulting flickering, the reduced quality of a downscaled video stream is also bound to affect the subjective experience. Adjustments to the compression ratio, resolution, or frame rate give rise to distinct visual artifacts, so it follows that they are not perceived in the same manner. When choosing an adaptation technique, providers benefit from knowing how acceptance can vary between the resulting quality reductions. Using the same subjective studies presented in Section 5.1, we ran further analyses with the ratings for presentations where the video quality was kept constant. Thus, we ignored the effect of quality switches (flicker) but kept the range of parameters for changes in resolution, frame rate, and compression ratios. These analyses are considered to be a prestudy for future investigations that will include quality adaptations in more than one dimension, as well as additional quality levels. Here, we present our initial suggestions for perceptually preferable quality adaptation schemes within the trade-offs recommended for mobile devices by Akamai [40], Apple [41], and Microsoft [42].

In this respect, Figure 12 depicts scores and statistics for the different quality adaptation schemes, arranged from the highest to lowest mean acceptance score. As seen from the figure, median and mean acceptance scores are below neutral for all adaptations with compression ratio at QP32 or above, frame rate at 10 fps or below, and resolution at 240×160 pixels or below. These findings imply that video quality adaptation at these levels is generally perceived as unacceptable for mobile devices with 480×320-pixel screens.

When it comes to frame rate, McCarthy et al. [43] suggested that 6 fps is sufficient for acceptable video quality, yet our data set does not provide support for this threshold. We found mean acceptance scores below neutral even at 15 fps. This decrease in acceptability scores could be related to the larger screens of today's mobile devices and possibly to an increase in the use and familiarity of watching mobile video. Judging from the implemented levels of compression and resolution and the results shown in Figure 12, we surmise that their thresholds in our setting are located around QP32 and 240×160 pixels. These acceptance thresholds for each adaptation technique define the lowest quality without a noteworthy reduction of average user satisfaction.

The only four levels with mean acceptance scores better than neutral are all different levels of compression adaptation, ranging from QP12 to QP28. Slightly below neutral follows frame rate adaptation at 15 fps. Going by these results, we can assume that QP compression is the adaptation technique that provides the most acceptable downscaled video quality. However, with severely limited bandwidth, these compression ratios may not yield sufficiently low bitrates, in which case it would be advisable to reduce the frame rate. Resolution adaptation appears to be the last resort, only to be applied under extremely poor conditions.

Furthermore, our results show that quality adaptations do not operate uniformly across video contents. We found both the spatial and temporal characteristics of different contents to interact with the applied adaptation technique. In the spatial domain, the quality acceptance for video contents with complex textural details was more negatively affected by resolution adaptations compared to contents low in spatial complexity. The quality ratings also seem to reflect a higher



(a) Liveness

(b) Number of dropped packets

Figure 10: Performance of different segment lengths.

visibility of compression artifacts in video with smooth or simple texture than in video with complex texture.

As for frame rate adaptation, videos with fast or unidirectional motion were rated lower than content with slow or nonorientable motion. In addition, people will likely not expect artificial movements to be as smooth as true-life movements. The interaction between compression artifacts and content characteristics may contribute to discrepancies in the actual acceptance of flicker for different video materials. With this in mind, it would be prudent for service providers to consider the type of video content before applying an adaptation technique.

All in all, for the best subjective experience, it is important to consider both the required downscaling and the type of content. With sufficient bandwidth available, compression adaptation is perceived to be more acceptable than both resolution and frame rate adaptation. However, if low bitrates are called for, or the content at hand is high in textural details, frame rate adaptation may be a more viable alternative.

5.3. Media Container Overhead at Low Bitrate Streaming. Mobile wireless networks like 3G typically have lower available bandwidths compared to wired networks. This means that video data must be available in lower bitrates for mobile devices. Regardless of which media container format is used, much of the overhead is proportional to the presentation unit rate, not the media bitrate. The presentation unit rate is usually (the exception is adaptation in the temporal domain, which involves reducing the bitrate through the presentation unit rate) constant across different quality levels, so this implies that the relative overhead of the container format is often higher for low bitrate videos. Consequently, the container overhead constitutes more of the stream bitrate in mobile scenarios characterized by low bitrate streams. In turn, less bandwidth is available for the audio and video data. Container stream overhead could be reduced by lowering the video frame rate or the audio sample rate, but, in our experience, this is rarely done.

The most common container formats used in adaptive bitrate streaming over HTTP are MPEG-2 Transport Streams (TS) [44] and the ISO base media file format (BMFF) [45] (often referred to as “fragmented MP4” when used in the context of segmented streaming). MPEG-2 TS is the container format used by Apple’s HTTP Live Streaming format [5]; it is popular on both iOS and Android devices, and, combined, these contribute to the vast majority of mobile streaming devices today. Both the MPEG DASH [6] and Microsoft Smooth Streaming systems [3] support the far more efficient fragmented MP4 container, but MPEG DASH has not yet been widely adopted by the industry.

In Figure 13, we plot the relative overhead of these containers as a function of the elementary stream bitrate for

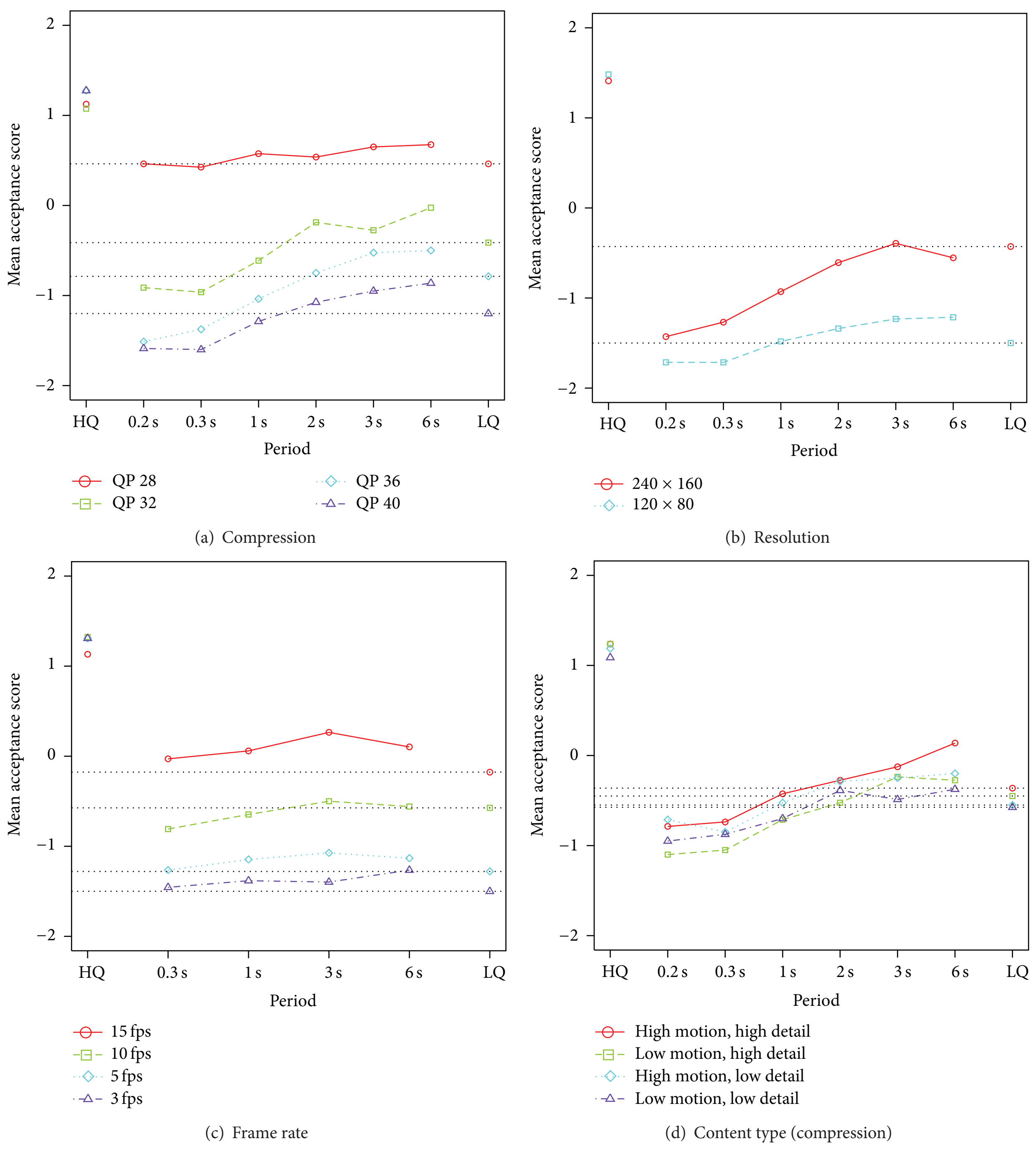


Figure 11: Mean acceptance scores for adaptation frequencies using (a) compression, (b) resolution, and (c) frame rate adaptation. (d) shows the impact of content type for the compression case.

a stream with 50 presentation units per second (same rate used for interlaced video in Europe). It is clear from this figure that fragmented MP4 has very little overhead. The overhead per presentation unit is only 32 bits when the contained media streams have fixed sample duration (fixed number of frames per second for video, or fixed number of samples per presentation unit for audio). We also see that, compared to the MP4 format, the relatively high overhead of MPEG-2 TS makes it unsuited for low bitrate streaming.

Another observation, not shown in Figure 13, is the low applicability of the MP4 format for low latency (live) streaming. MP4 is optimized for random access; therefore, it has a mandatory index where the byte offset to every frame is stored. Because the index can only be written after the encoded size of every frame in the segment is known, MP4 carries with it a delay equal to the segment duration. However, in adaptive streaming, each segment typically contains only a single random access point (a keyframe) at the beginning of

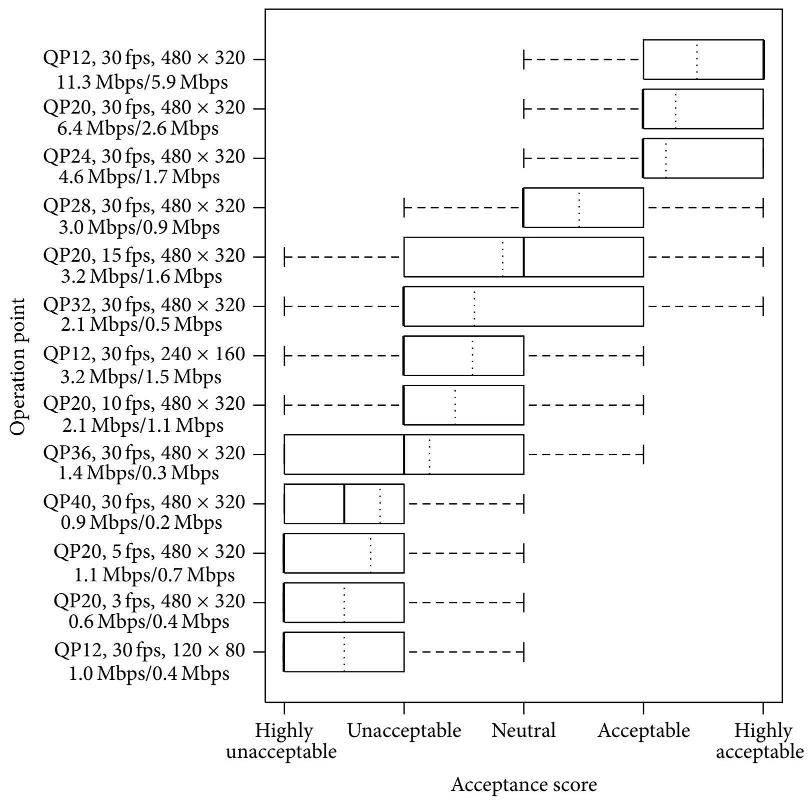


Figure 12: Box plot of acceptance scores for compression, resolution, and frame rate adaptations. The central box spans the interquartile range, with minimum and maximum scores illustrated by "whiskers" to the left and right. Within the box, the bold line corresponds to the median, whereas the dotted line represents the mean acceptance score. The resulting bitrates are also included for each step. The first bitrate is when using I-frames only, which is used in the subjective assessments in order to maintain focus on the given quality parameters and avoid irrelevant artifacts. A real-world scenario would include interframe coding (like IBB∗ used in the second bitrate) giving a lower rate (we did not observe any visual difference between the I∗ and IBB∗ videos); these rates are comparable to the rates observed in the Comoyo analysis given in Section 3.

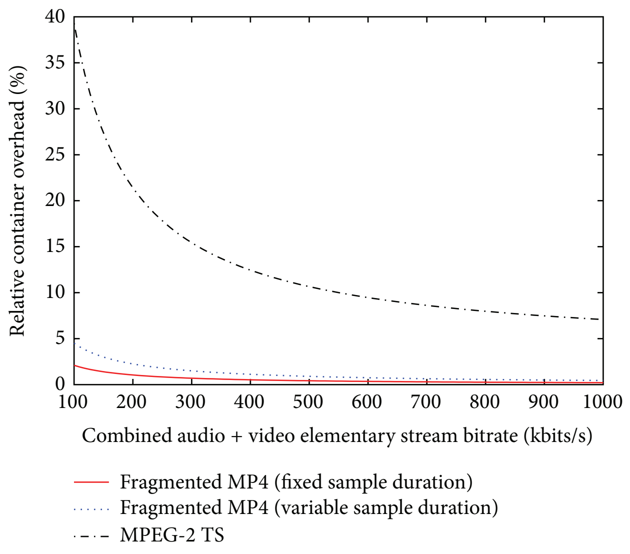


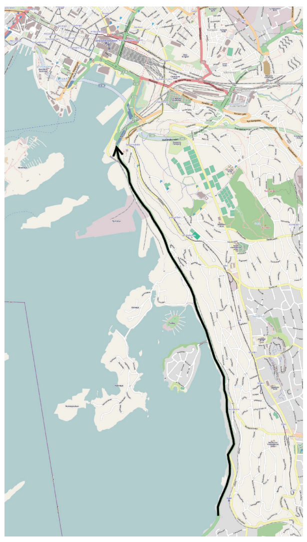
Figure 13: Relative media container overhead as a function of the elementary stream bitrate in a stream with 50 presentation units per second.

a segment (typically two seconds in duration). Accordingly, random access within a segment is pointless. Instead of an index, live streaming latency can be reduced by using a container format that precedes each frame with its encoded size in bytes. This way, the segment can be transmitted while it is being encoded, and the receiver can access the data concurrently [46].

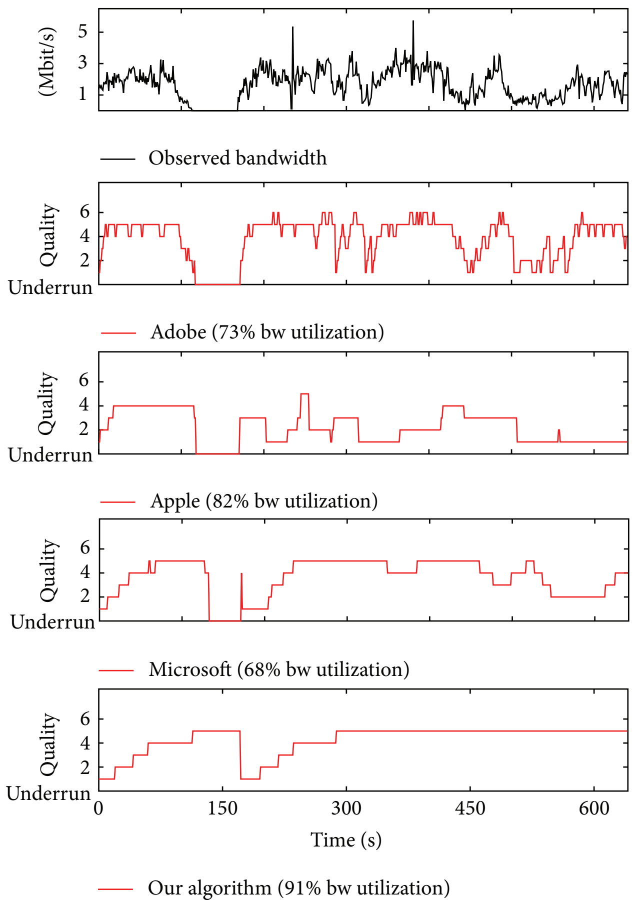
6. Quality Adaptation Schemes

In Section 3, we discussed frequent quality changes and playout stalls due to buffering. Efficient quality adaptation schemes are essential for avoiding quality degradations caused by fluctuating network availability. These investigations were performed in wired networks. However, the network conditions for mobile devices are very different.

Therefore, in order to develop an adaptation algorithm for the mobile scenario, we have performed a comparison of commercial adaptive HTTP streaming solutions in commercial 3G networks [13] using six discrete quality levels (ranging from 250 to 3000 kbit/s). For every segment downloaded



(a)



(b)

Figure 14: A comparison of quality adaptation algorithms in different media players. The map on the left shows the used bus route. For more examples, see [10].

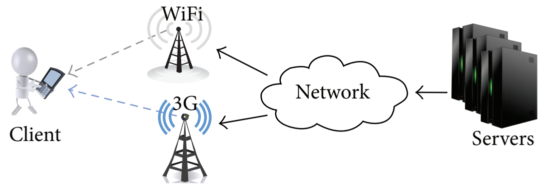


Figure 15: A client device equipped with multiple network interfaces which has several active network connections.

for a given streaming client, Figure 14 plots the quality level index as a function of time on a bus route in Oslo (Norway). Large differences between the tested systems can be observed, and our experiments show that the existing solutions all have shortcomings like frequent switches and playout stalls. Apple's and Adobe's players represent two opposites. Apple's player [5] aims to avoid buffer underruns at all costs, resulting in low average quality. This means that Apple sacrifices high average quality for stable quality. In Figure 14, Apple's player uses most of the available bandwidth, but, due to the pessimistic behavior, downloads of many high-quality segments are started but later stopped in favor of a low quality segment (thus, wasting a lot of bandwidth). Adobe [4], on the other hand, strictly follows the available bandwidth. The player always picks the quality level that most closely matches the current bandwidth. This leads to rapid oscillations in quality and almost no protection against buffer underruns (since the buffer is usually empty). The best performer among the commercial media players in our mobile streaming scenario is Microsoft's player [3]. It has fairly good average quality and not too frequent switches between quality levels. Thus, Microsoft's solution falls somewhere between Apple and Adobe, but there is still potential for better utilization of the available bandwidth, a reduction of quality changes and underruns.

In this respect, based on our investigations [10], potential new quality adaptation algorithms for mobile scenarios can be improved using the following recommendations.

(a) Choose Quality Layers Conservatively While Filling the Buffer. To avoid buffer underruns, the quality scheduler should limit quality selection based on the estimated available bandwidth until the buffer is sufficiently full. In other words,

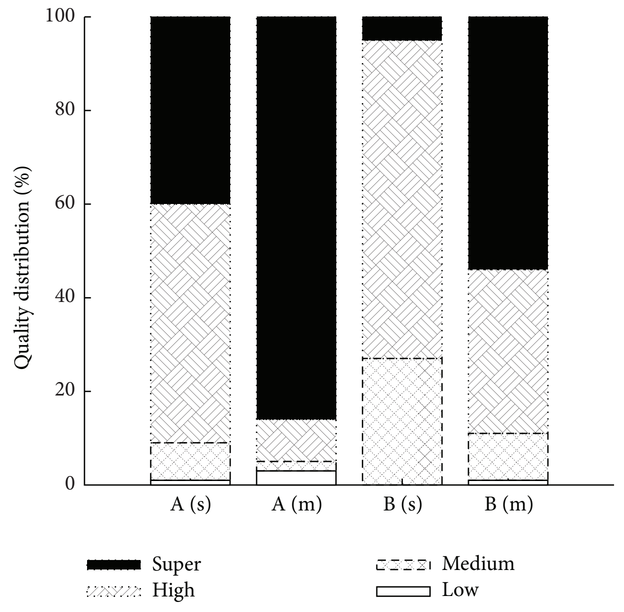


FIGURE 16: Quality distribution for different types of streaming in real-world networks. A: on-demand streaming, B: live streaming; (s) is single link; (m) is multilink.

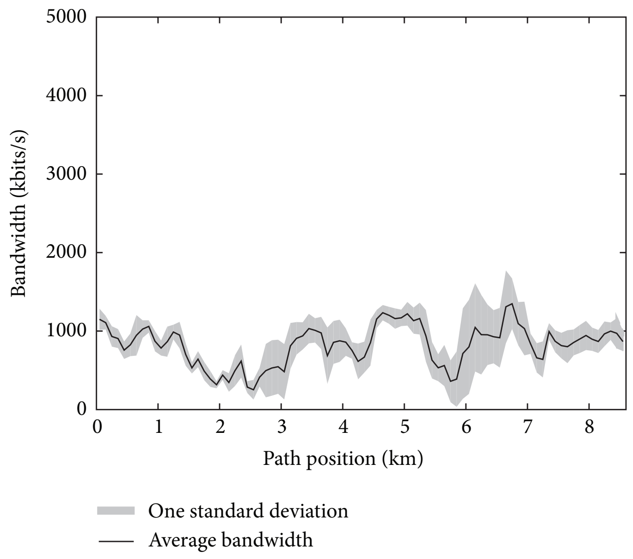


FIGURE 17: Average measured bandwidth along the tram commute path with standard deviation over multiple measurements.

when the buffer fill level is low, the quality scheduler should try to avoid draining the buffer by only picking quality levels whose bitrates are slightly lower than the estimated download bandwidth.

(b) Sample Network Throughput More Frequently Than Once per Segment, and Estimate by Moving Average of Samples. When estimating the download bandwidth, an exponentially weighted moving average of several recent measurements that are sampled more frequently than once per segment reduces the impact of observations made from a single segment's download time. This smoothens out the rapid bandwidth fluctuations that could otherwise occur and reduces unnecessary oscillations in quality.

(c) Prepare for Temporary Network Outages. This recommendation implies that larger buffers should be used so that data can be available for longer outages. This means that we, for example, can use the available bandwidth above the playout rate (or trade off some quality) to prevent buffer underruns, have a more stable video quality, and continue playback, even during network outages.

(d) Require Longer Prefetched Times for Higher Quality Layers. The buffer fullness thresholds for switching between quality levels should be scaled according to the bitrate difference between levels. Since the visual quality gain increases approximately logarithmic with the bandwidth invested, requiring a longer temporal buffer for higher quality layers emphasizes the reduced quality gain of consuming bandwidth for a higher quality layer compared to that of ensuring long-term availability of lower quality layers.

(e) Establish Asymmetric Thresholds for Switching Up and Down. The thresholds for switching between quality levels should take into account whether the quality switch is towards lower or higher quality.

(f) Prevent Switching Up Right after Reducing Quality. After a drop in quality, the quality scheduler should for a short period prohibit switches to higher qualities. This reduces the number of quality fluctuations.

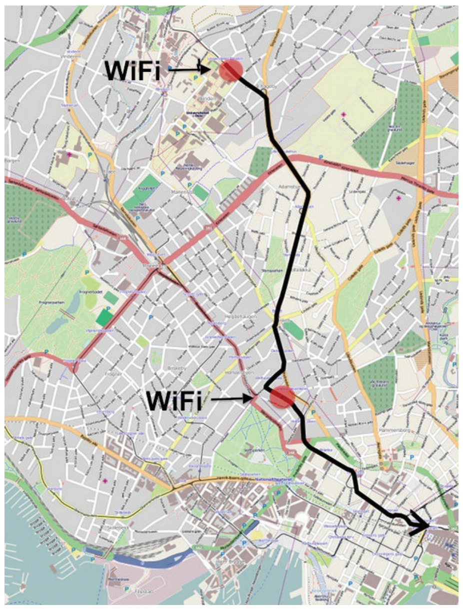
Our implementation of these recommendations is Algorithm 1, a buffer-based reactive algorithm.

Algorithm 1 (reactive algorithm). The buffer-based reactive algorithm selects the video bitrate based on the number of seconds of video that are preloaded in the buffer. Given the average bitrate R_i for quality layer i of a video and B a number of seconds we want to buffer, we establish the requirement $T_N = B \cdot (R_N - R_1)/(R_2 - R_1)$ for quality layer N, where $B = 10$ s.

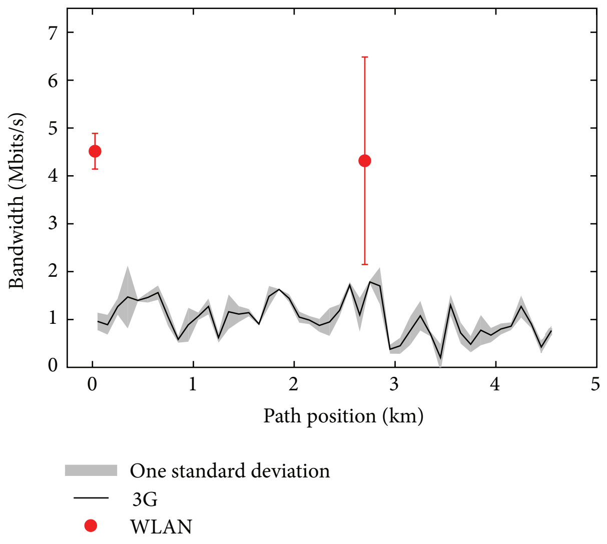
The algorithm starts always at quality layer 1 and increases in steps of 1 layer to layer i if $1.2T_i$ are buffered and decreases immediately to layer j if the buffer falls to T_j. After a quality drop, increasing is blocked for $2B$.

For better protection against oscillations and playout interruptions, the quality level is capped to a level i if R_i is the highest rate that is supported by the recently available bandwidth $r(t)$. It is computed as $r(t) = 0.9r(t-1) + 0.1r_t$, where r_t is the last 1-second sample.

To experimentally evaluate the quality differences between the different algorithms, we performed video streaming experiments on various commute routes in Oslo (Norway) using bus, tram, underground, and ferry (recorded datasets are available [8]). In Figure 14, we have implemented Algorithm 1 and evaluated its performance in a mobile scenario, denoted by "our algorithm" in the last plot. When using this algorithm, we found the performance



(a)



(b)

Figure 18: Observed 3G and WiFi download rates while traveling by tram in Oslo. WiFi was only available at the marked spots.

with respect to quality scheduling to be most similar to Microsoft's algorithm. However, the figure also shows that we achieve better protection against buffer underruns due to a larger buffer, more intelligent quality switches, and better bandwidth utilization. This resulted in higher quality of experience for the users. Nevertheless, as the number of users streaming video to mobile devices increases, the competition for the scarce network resources also increases. In our real-world experiments [10], we did not observe many competing users in the commute vehicles. In theory, the recommendations presented above should improve the situation in this scenario too since it targets bitrate oscillation problems, but, as shown for wired networks, the commercial algorithms struggle to share resources in a stable and fair manner [17, 47]. Thus, new experiments with a large number of concurrent users should be performed to see if further adjustments need to be done.

7. Bandwidth Improvements Using Multilink

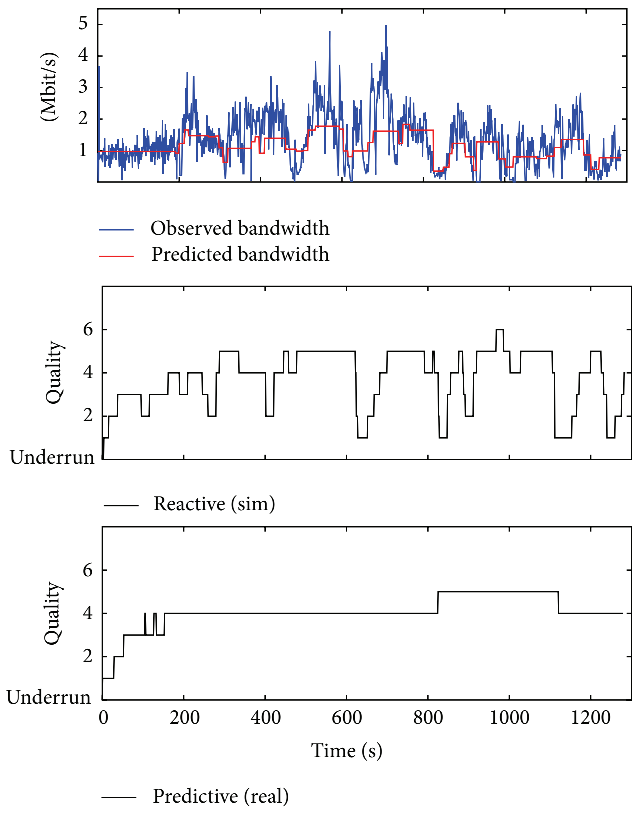
Wireless networks often provide unreliable and low bandwidths, especially when users are on the move. One way to alleviate this problem is to increase the available bandwidth by aggregating multiple physical links into one logical link. Such a solution would be available to a large share of users, as most mobile devices on the market today are multihomed. For example, smartphones and tablets are equipped with both WLAN and 3G/4G interfaces, as shown in Figure 15.

In a series of steps, we implemented a solution for multilink bandwidth aggregation in order to increase the throughput of data transfer over HTTP [48]. The first step involved modifying our streaming client and adjusting the algorithms for adaptive streaming. Secondly, by dividing video segments into smaller, logical subsegments, with the range retrieval request-feature of HTTP/1.1, it was possible to request specific parts of a file. The subsegment requests were then distributed across the available interfaces, with the size of each subsegment determined by the estimated link capacity.

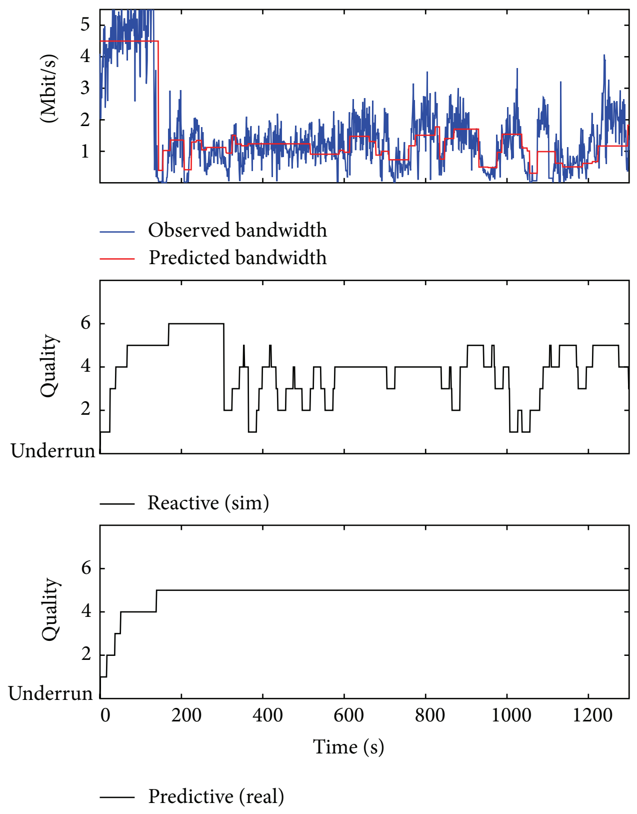
The size of a subsegment has large impact on performance; if a slow link is allocated an excessively large share of a segment, performance might be worse than for a single link solution. For example, the segment may not be ready when it is supposed to be played out, causing a deadline miss and playback interruption. For further improvements in performance, we used HTTP pipelining to minimize the idle time of a link. Subsegment size and request distribution algorithms are discussed in more detail in [49].

Several experiments were run in order to evaluate the potential gain of bandwidth aggregation in the context of adaptive video streaming, with performance evaluated for both on-demand and live streaming. Our client devices were connected to public wireless networks, as well as fully controlled networks, where we introduced different levels of bandwidth and latency heterogeneity. The measured performance showed a substantial quality increase with bandwidth aggregation, along with a drop in the number of playout stalls [49].

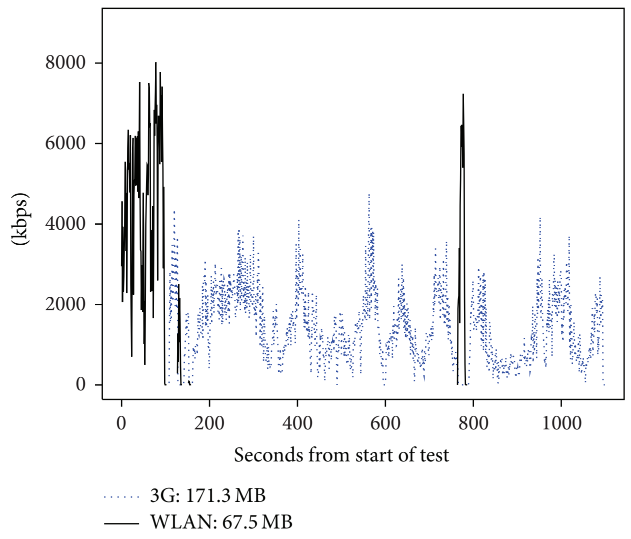
The potential gain in terms of video segment quality of an example experiment is shown in Figure 16. Here, the mobile device was concurrently connected to both WLAN and 3G networks where the average throughput was measured to be 287 kB/s and 167 kB/s and the average RTT to be 30 ms and 220 ms, respectively, for the two types of networks. Each segment consisted of two seconds of video (following findings presented in Section 5.1). For on-demand streaming, a buffer and startup delay of two segments were used. With live



(a) The predictive quality adaptation algorithm versus the reactive algorithm in a 3G network



(b) The predictive quality adaptation algorithm versus the reactive algorithm when switching between WLAN and 3G



(c) Example of achieved throughput and handovers in the network switching scenario

Figure 19: Performance of a reactive algorithm implemented as proposed in Section 6 versus a predictive algorithm using bandwidth geolookups. The experiments are performed using both a 3G scenario and a network switching scenario as described in Section 7. To compare the algorithms on equal terms, one is tested in the real-world setting and the other is simulated using the exact same bandwidth trace. Experiments where the simulated and the real-world algorithms are switched are presented in [10, 11] with similar results.

streaming, there was no buffer and segments were skipped if the client lagged too far behind the broadcast. As shown in the figure, when we added the second link, the number of requested and downloaded high-quality segments was at least doubled; moreover, we observed significantly fewer playout stalls compared to the fastest of the single links.

From this, we see that bandwidth aggregation can be used to increase the performance of video streaming on mobile devices, provided that the scheduling of segments over the different networks is correctly implemented, that is, taking into account the characteristics of the different interfaces. However, bandwidth aggregation comes with a cost, such as reduced battery life. We are currently working on a more dynamic aggregation approach, where the extra link(s) will be enabled only when needed.

8. Bandwidth Prediction

In Figure 14, we showed that there are large differences between quality adaptation algorithms for on-demand scenarios. However, with a few changes, the quality of experience can be significantly improved. With our enhanced algorithm, we touched on the concept of bandwidth prediction using an exponentially weighted moving average. This was a very short-term prediction, only to be used for the next segment to be downloaded. However, if an accurate long-term prediction would be possible, for example, while streaming on a commute route, the buffering and quality adaptation choices could be greatly improved. Looking at our bandwidth measurements for the commute routes in Oslo, for example, the tram in Figure 17, we see that the observed bandwidth at a given location can be fairly predictable as the different measurements have a very little variance. Thus, if this can be used for long-term predictions, the likelihood of buffer underruns can be reduced, and we can smooth out the quality because we have a larger buffer time window to cancel out bandwidth fluctuations and outages. For example, in a commute scenario, we may easily collect information about the following:

(i) the duration of the streaming session, for example, how much time the tram takes from A to B (this can easily be logged for repeated trips, or retrieved from public traffic services),

(ii) the geographical position as a function of time for the duration of the streaming session (e.g., through positioning data recorded on previous streaming sessions on a receiver equipped with a GPS or similar device),

(iii) the bandwidth for a given geographical position, for example, building a bandwidth lookup database through crowd-sourcing, where the video application reports back its position and achieved bandwidth.

Commute routes are usually highly deterministic, with respect to both geographical path and duration. When streaming video while commuting, this kind of long-term planning is possible using a location-based bandwidth lookup service for bitrate planning [10, 50]. Subsequently, Singh et al. [51] proposed a similar geopredictive service as a network coverage map service.

To evaluate such a service, we built a time-location-bandwidth database for multiple commute routes and used this for long-term planning of adaptive HTTP streaming sessions. Our predictive quality adaptation algorithm calculates the predicted amount of data along the path and downloads segments in a quality according to the average bitrate; that is, the highest (stable) quality level that does not result in a buffer underrun is selected. To cope with prediction errors due to, for example, network congestion, the predictive algorithm is combined with a reactive algorithm based on the recommendations in Section 6. The predictive algorithm is explained in Algorithm 2.

Algorithm 2 (predictive algorithm). The predictive algorithm requires a planned commuting route as input. It then queries the location-based bandwidth lookup service for predictions along the planned route in samples of 100 meters.

Based on the query response, the client calculates a schedule that selects for every subsequent segment the highest quality level that could be used for the rest of the trip without any buffer underrun; that is, it builds an increasing step function of quality layers.

Segments are downloaded in playout order. For each downloaded video segment, the client measures and logs throughput, current position, and buffer fill level. If buffer fill levels are lower than planned, it compares with the reactive algorithm (Algorithm 1 without the cap) and selects the lower quality layer of either planned layer or layer chosen by the reactive algorithm. The client reports its samples to the lookup service in batches.

For every segment to be downloaded, the results of the reactive and predictive algorithms are compared, and the lowest quality level is chosen. The combination of these two algorithms gives a more stable quality. The predictive algorithm prevents the reactive algorithm from scaling up the quality too soon, while the reactive algorithm prevents buffer draining. Finally, in order to support deviations from the predicted path and travel duration, as well as live streaming, our system recalculates the adaptation plan for every segment downloaded. By doing this, we are continually updating the adaptation parameters (buffer fill level, current bandwidth, geographical position, current time, etc.), which allows the adaptation plan to self-correct as we are progressing along our travel path.

In our real-world experiments, again using public transportation in Oslo, we used a commercial 3G network for downloading data and combined this with a WiFi network (Eduroam) where this was available along the route. The bandwidth measurements for one of several routes that we used in our experiments are presented in Figure 18. We traveled the route, which leads from the main university campus in Oslo to the city center, by tram. The 3G network was available the entire path whereas WiFi was only available in proximity of the University of Oslo and Oslo University College. We see from the bandwidth plot that the 3G download rate in a particular location is highly predictable, as the variance in observations is quite small. The variance for the second WLAN spot is slightly higher as the tram goes by the access point at speed, and the time to connect and the signal strength varied between the experiments.

Figure 19(a) compares our predictive algorithm (the combination of the reactive and the predictive algorithms as described above) with our reactive algorithm (described in Section 6). To be able to directly compare the two quality adaptation algorithms on exactly the same bandwidth data, one of the two results had to be simulated based on observed bandwidth. We can see that the quality is significantly more stable with the predictive algorithm. Moreover, we avoid the visually disruptive quality jumps [38] that the reactive algorithm had to make to avoid buffer underrun.

Looking at Figure 18, we also see that other networks were available along the path. As a further enhancement, we combined the predictive adaptation algorithms with the multilink solution presented in Section 7 [11]. Multiple test runs were performed to make sure the system discovered areas with higher bandwidth networks. Figure 19(b) shows an example of the video quality when the client switched between networks (selecting the predicted best one), using the same tram ride and video. The high-speed WLAN was available at the start of the ride, which resulted in a significant higher video quality than with only 3G. With the predictive scheduler, the media player was allowed to stream at quality level 5 (1500 kbps) for most of the trip, compared to level 4 (1000 kbps) when only 3G was used. The higher bandwidth of the WLAN enabled the client to receive more data, building up a bigger buffer and requesting data in a higher quality. With respect to handover performance, we have plotted the throughput for the streaming sessions from Figure 19(b) in Figure 19(c). From the plots, we can observe that the handover time is minimal and that the client receives data without significant idle periods. These results show the potential of combining transparent handover with a location-based, adaptive video streaming system.

However, there are still many challenges to solve. The most important of these are the following.

(i) Wireless links have a major influence on round-trip times. Depending on configuration, 3G networks may suffer from considerable buffer bloat, while this has not been observed for WiFi access networks. This can lead to delay variances between 3G and WiFi networks on the scale of several seconds.

(ii) We perform handover to WiFi when it is available. However, authentication takes time, and a moving receiver may leave coverage after a very short period. Since throughput on the WiFi link depends strongly on proximity to the base station, it might actually be situation-dependent whether 3G or WiFi yields the higher data rate within the WiFi coverage area.

(iii) Our prediction method depends on a predictable vehicle speed, because frequent GPS measurements drain the receiver's battery. However, neither the history of tram movement nor the public transport company's real-time update system provides enough details for predicting WiFi coverage strength. This information may be acquired from a 3G positioning system.

9. Discussion

With this paper, we have summarized development steps that have led to the development of our algorithm for predictive streaming to wireless receivers over multiple access networks. In a market where HTTP adaptive streaming increasingly dominates the streaming infrastructure, we based this work exclusively on this kind of streaming system.

We argue based on existing work that the currently applied rule of thumb is still valid, which favors long-term stable quality as long as buffer underrun events can be avoided. However, we acknowledge that recent studies show that the situation is not quite as simple for lower bitrates and thus requires more research. For this work, we chose to aim for long-term constant quality in HTTP adaptive streaming in spite of this.

Although there are frequent discussions about the need for live streaming over an HTTP adaptive streaming infrastructure, we found in analyzing traces of a commercial provider that this user requirement is commercially relevant and that it leads to an undesirable number of buffer underruns and bitrate switches in clients. To understand this situation better, we investigated the interaction between HTTP adaptive streaming and TCP in a bottleneck situation where a big number of HTTP adaptive streams competed with each other. We found that a variety of application-layer methods can reduce this competition, but we could not avoid transient congestion without modifying mechanisms in the transport layer. An option at the transport layer that we proposed in this paper relies on congestion window limitations; other promising approaches could be found in work by Esteban et al. [37] and work by Nazir et al. [52]. These results promise that the transport layer can interact in beneficial ways with HTTP adaptive streaming, but the interaction with other kinds of traffic needs to be investigated in future research.

At the application layer, we showed in terms of interaction with TCP that long (10-second) segments are not more beneficial than short (2-second) segments. We could also conclude that 2-second segments are sufficient for avoiding the fact that users perceive quality changes as flicker, thereby avoiding severe quality reduction. Looking at multiple scaling dimensions, we found that quantization strength is the means of reducing quality and leads to weaker quality reduction than the other scaling dimensions and could thereby develop an application-layer adaptation strategy.

The first strategy that we presented in this work was a client-side reactive algorithm that is conservative in its avoidance of buffer underruns and trying to avoid quality switches. We compared these results with the algorithms found in commercial players, which is the typical approach in related work. The abundance of existing research proposals would warrant a comparison among them, but, in this work, we aimed instead at an improvement of our algorithm under the assumption of two additional infrastructure elements:

multiple access networks and a centralized bandwidth lookup service.

We developed a predictive algorithm for HTTP adaptive streaming that interacts with a bandwidth lookup service by planning bandwidth for well-known commuting routes. Our approach combines this with the knowledge of available bandwidth in different networks and can plan handover between them to achieve the best possible plan for HTTP adaptive streaming. This field of research is highly promising, but our results are of course limited to routes that can be preplanned, whereas an exploitation of a bandwidth lookup service for arbitrary movements of the receiver would be desirable. Furthermore, energy efficiency is a limitation of this scheme and should therefore be a topic of future research as well.

10. Conclusion

Adaptive HTTP streaming is frequently used to deliver video to mobile devices. However, compared to fixed connections, the bandwidth in mobile broadband networks fluctuates more. Also, mobile devices are more heterogeneous than, for example, TV sets and desktop computers, for example, with respect to processor, screen size, and resolution. In this paper, we have presented the research steps that we have undertaken so far towards a solution for HTTP adaptive streaming to wireless receivers that can make use of multiple wireless networks and use a bandwidth lookup service to plan network availability. While this work presents a considerable number of results that have advanced the state of the art, we present also a variety of open questions that range from challenges in understanding QoE in HTTP adaptive streaming scenarios to prediction of resource availability for freely moving wireless receivers.

Conflict of Interests

The authors declare that there is no conflict of interests regarding the publication of this paper.

References

[1] YouTube, "YouTube statistics," November 2012, http://www.youtube.com/t/press_statistics.

[2] Cisco, "Cisco visual networking index: global mobile data traffic forecast update, 2013–2018," http://www.cisco.com/c/en/us/solutions/collateral/service-provider/visual-networking-index-vni/white_paper_c11-520862.html.

[3] A. Zambelli, "Smooth streaming technical overview," 2009, http://www.iis.net/learn/media/on-demand-smooth-streaming/smooth-streaming-technical-overview.

[4] Adobe, "HTTP dynamic streaming on the Adobe Flash platform," 2010, https://bugbase.adobe.com/index.cfm?event=file.view&id=2943064&seqNum=6&name=httpdynamic-streaming_wp_ue.pdf.

[5] R. Pantos, J. Batson, D. Biderman, B. May, and A. Tseng, "HTTP live streaming," 2010, http://tools.ietf.org/html/draft-pantos-http-live-streaming-04.

[6] T. Stockhammer, "Dynamic adaptive streaming over HTTP: standards and design principles," in *Proceeding of the 2nd Annual ACM Multimedia Systems Conference (MMSys '11)*, pp. 133–144, New York, NY, USA, February 2011.

[7] Move Networks, "Internet television: challenges and opportunities," Tech. Rep., Move Networks, 2008.

[8] H. Riiser, P. Vigmostad, C. Griwodz, and P. Halvorsen, "Commute path bandwidth traces from 3G networks: analysis and applications," in *Proceedings of the 4th ACM Multimedia Systems Conference (MMSys '13)*, pp. 114–118, March 2013.

[9] C. Müller, S. Lederer, and C. Timmerer, "An evaluation of dynamic adaptive streaming over HTTP in vehicular environments," in *Proceedings of the 4th Workshop on Mobile Video (MoVid '12)*, pp. 37–42, February 2012.

[10] H. Riiser, T. Endestad, P. Vigmostad, C. Griwodz, and P. Halvorsen, "Video streaming using a location-based bandwidthlookup service for bitrate planning," *ACM Transactions on Multimedia Computing, Communications and Applications*, vol. 8, no. 3, pp. 24:1–24:19, 2012.

[11] K. Evensen, A. Petlund, H. Riiser et al., "Mobile video streaming using location-based network prediction and transparent handover," in *Proceedings of the International Workshop on Network and Operating Systems Support for Digital Audio and Video (NOSSDAV '11)*, pp. 21–26, ACM, New York, NY, USA, June 2011.

[12] S. Akhshabi, A. C. Begen, and C. Dovrolis, "An experimental evaluation of rate-adaptation algorithms in adaptive streaming over HTTP," in *Proceedings of the 2nd Annual ACM Multimedia Systems Conference (MMSys '11)*, pp. 157–168, February 2011.

[13] H. Riiser, H. S. Bergsaker, P. Vigmostad, P. Halvorsen, and C. Griwodz, "A comparison of quality scheduling in commercial adaptive HTTP streaming solutions on a 3G network," in *Proceeding of the 4th Workshop on Mobile Video (MoVid '12)*, pp. 25–30, New York, NY, USA, February 2012.

[14] M. Zink, O. Künzel, J. Schmitt, and R. Steinmetz, "Subjective impression of variations in layer encoded videos," in *Proceedings of the International Workshop on Quality of Service*, pp. 137–154, 2003.

[15] S. Tavakoli, K. Brunnström, K. Wang, B. Andren, M. Shahid, and N. Garcia, "Subjective quality assessment of an adaptive video streaming model," in *Image Quality and System Performance XI*, S. Triantaphillidou and M.-C. Larabi, Eds., vol. 9016 of *Proceedings of SPIE*, 2014.

[16] A. Borowiak and U. Reiter, "Long duration audiovisual content: Impact of content type and impairment appearance on user quality expectations over time," in *Proceedings of the 5th International Workshop on Quality of Multimedia Experience (QoMEX '13)*, pp. 200–205, July 2013.

[17] S. Akhshabi, S. Narayanaswamy, A. C. Begen, and C. Dovrolis, "An experimental evaluation of rate-adaptive video players over HTTP," *Signal Processing: Image Communication*, vol. 27, no. 4, pp. 271–287, 2012.

[18] J. Jiang, V. Sekar, and H. Zhang, "Improving fairness, efficiency, and stability in HTTP-based adaptive video streaming with FESTIVE," in *Proceedings of the 8th ACM International Conference on Emerging Networking EXperiments and Technologies (CoNEXT '12)*, pp. 97–108, ACM Press, New York, NY, USA, December 2012, http://dl.acm.org/citation.cfm?doid=2413176.2413189.

[19] S. Akhshabi, L. Anantakrishnan, C. Dovrolis, and A. C. Begen, "Server-based traffic shaping for stabilizing oscillating adaptive streaming players," in *Proceeding of the 23rd ACM Workshop on Network and Operating Systems Support for Digital Audio and*

Video (NOSSDAV '13), pp. 19–24, New York, NY, USA, February 2013.

[20] K. Miller, E. Quacchio, G. Gennari, and A. Wolisz, "Adaptation algorithm for adaptive streaming over HTTP," in *Proceedings of the 19th International Packet Video Workshop (PV '12)*, pp. 173–178, May 2012.

[21] Z. Li, X. Zhu, J. Gahm et al., "Probe and adapt: rate adaptation for HTTP video streaming at scale," *IEEE Journal on Selected Areas in Communications*, vol. 32, no. 4, pp. 719–733, 2014.

[22] R. Houdaille and S. Gouache, "Shaping HTTP adaptive streams for a better user experience," in *Proceedings of the 3rd Multimedia Systems Conference (MMSys '12)*, pp. 1–9, ACM Press, New York, NY, USA, 2012.

[23] W. Pu, Z. Zou, and C. W. Chen, "Video adaptation proxy for wireless Dynamic Adaptive Streaming over HTTP," in *Proceedings of the 19th International Packet Video Workshop (PV '12)*, pp. 65–70, IEEE, May 2012.

[24] A. Mansy, M. Ammar, J. Chandrashekar, and A. Sheth, "Characterizing client behavior of commercial mobile video streaming services," in *Proceedings of the Workshop on Mobile Video Delivery (MoViD '14)*, vol. 8, ACM, New York, NY, USA, 2014.

[25] M. Siekkinen, M. A. Hoque, J. K. Nurminen, and M. Aalto, "Streaming over 3G and LTE: how to save smartphone energy in radio access network-friendly way," in *Proceedings o f the 5th Workshop on Mobile Video (MoVid '13)*, pp. 13–18, ACM Press, New York, NY, USA, 2013.

[26] D. Havey, R. Chertov, and K. Almeroth, "Receiver driven rate adaptation for wireless multimedia applications," in *Proceedings of the 3rd ACM Multimedia Systems Conference (MMSys '12)*, pp. 155–166, February 2012.

[27] K. Evensen, T. Kupka, D. Kaspar, P. Halvorsen, and C. Griwodz, "Quality-adaptive scheduling for live streaming over multiple access networks," in *Proceeding of the 20th ACM Workshop on Network and Operating System Support for Digital Audio and Video (NOSSDAV '10)*, pp. 21–26, New York, NY, USA, June 2010.

[28] O. Oyman and S. Singh, "Quality of experience for HTTP adaptive streaming services," *IEEE Communications Magazine*, vol. 50, no. 4, pp. 20–27, 2012.

[29] T. Kupka, C. Griwodz, P. Halvorsen, D. Johansen, and T. Hovden, "Analysis of a real-world HTTP segment streaming case," in *Proceedings of the 11th European Conference on Interactive TV and Video (EuroITV '13)*, pp. 75–84, June 2013.

[30] L. Kontothanassis, "Content delivery considerations for different types of internet video," in *ACM Multimedia Systems Conference (MMSys '11)*, San Jose, Calif, USA, February 2011.

[31] H. Jiang, Y. Wang, K. Lee, and I. Rhee, "Tackling bufferbloat in 3G/4G networks," in *Proceedings of the ACM Internet Measurement Conference (IMC '12)*, pp. 329–342, November 2012.

[32] T. Kupka, P. Halvorsen, and C. Griwodz, "Performance of on-off traffic stemming from live adaptive segmented HTTP video streaming," in *Proceedings of the 37th Annual IEEE Conference on Local Computer Networks (LCN '12)*, pp. 401–409, October 2012.

[33] L. Plissonneau and E. Biersack, "A longitudinal view of HTTP video streaming performance," in *Proceeding of the 3rd ACM Multimedia Systems Conference (MMSys '12)*, pp. 203–214, New York, NY, USA, February 2012.

[34] L. A. Grieco and S. Mascolo, "Performance evaluation and comparison of Westwood+, New Reno, and Vegas TCP congestion control," *ACM SIGCOMM Computer Communication Review*, vol. 34, no. 2, pp. 25–38, 2004.

[35] S. Ha, I. Rhee, and L. Xu, "CUBIC: a new TCP-friendly high-speed TCP variant," *SIGOPS Operating Systems Review*, vol. 42, pp. 64–74, 2008.

[36] A. Aggarwal, S. Savage, and T. Anderson, "Understanding the performance of TCP pacing," in *Proceedings of the IEEE INFOCOM Conference*, pp. 1157–1165, Tel-Aviv, Israel, March 2000.

[37] J. Esteban, S. A. Benno, A. Beck, Y. Guo, V. Hilt, and I. Rimac, "Interactions between HTTP adaptive streaming and TCP," in *Proceedings of the 22nd international workshop on Network and Operating System Support for Digital Audio and Video (NOSSDAV '12)*, pp. 21–26, ACM, New York, NY, USA, June 2012.

[38] P. Ni, R. Eg, A. Eichhorn, C. Griwodz, and P. Halvorsen, "Flicker effects in adaptive video streaming to handheld devices," in *Proceeding of the 19th ACM International Conference on Multimedia ACM Multimedia (MM '11)*, pp. 463–472, New York, NY, USA, December 2011.

[39] ITU-T P.910, "Subjective video quality assessment methods for multimedia applications," International Telecommunications Union, 1999.

[40] Akamai, "Akamai HD for iPhone encoding best practices," 2010, http://www.akamai.com/dl/akamai/iphone_wp.pdf.

[41] Apple, *Using HTTP live streaming*, https://developer.apple.com/library/ios/documentation/networkinginternet/conceptual/streamingmediaguide/UsingHTTPLiveStreaming/UsingHTTPLiveStreaming.html.

[42] Microsoft, "Delivering live and on-demand smooth streaming," http://download.microsoft.com/download/4/E/5/4E599FBB-6E34-4A74-B3C5-1391CB0FD55F/Delivering_Live_and_On-Demand_Smooth_Streaming.pdf.

[43] J. D. McCarthy, M. A. Sasse, and D. Miras, "Sharp or smooth ?: comparing the effects of quantization vs. frame rate for streamed video," in *Proceedings of the Conference on Human Factors in Computing Systems (CHI '04)*, pp. 535–542, April 2004.

[44] ISO/IEC, "Generic coding of moving pictures and associated audio information: systems," ISO/IEC 13818-1:2007, 2007, http://www.iso.org/iso/catalogue_detail?csnumber=44169.

[45] ISO/IEC 14496-12, "Coding of audio-visual objects—Part 12: ISO base media file format," 2004, http://www.iso.org/iso/iso_catalogue/catalogue_tc/catalogue_detail.htm?csnumber=38539.

[46] H. Riiser, P. Halvorsen, C. Griwodz, and D. Johansen, "Low overhead container format for adaptive streaming," in *Proceeding of the ACM SIGMM Conference on Multimedia Systems (MMSys '10)*, pp. 193–198, New York, NY, USA, February 2010.

[47] S. Akhshabi, L. Anantakrishnan, C. Dovrolis, and A. C. Begen, "What happens when HTTP adaptive streaming players compete for bandwidth?" in *Proceedings of the 22nd ACM Workshop on Network and Operating Systems Support for Digital Audio and Video (NOSSDAV '12)*, pp. 9–14, June 2012.

[48] D. Kaspar, K. R. Evensen, P. E. Engelstad, and A. F. Hansen, "Using HTTP pipelining to improve progressive download over multiple heterogeneous interfaces," in *Proceedings of the IEEE International Conference on Communications (ICC '10)*, pp. 1–5, Cape Town, South Africa, 2010.

[49] K. Evensen, D. Kaspar, C. Griwodz, P. Halvorsen, A. F. Hansen, and P. Engelstad, "Using bandwidth aggregation to improve the performance of quality-adaptive streaming," *Signal Processing: Image Communication*, vol. 27, no. 4, pp. 312–328, 2012.

[50] H. Riiser, P. Vigmostad, C. Griwodz, and P. Halvorsen, "Bitrate and video quality planning for mobile streaming scenarios using a GPS-based bandwidth lookup service," in *Proceeding of the 12th IEEE International Conference on Multimedia and Expo (ICME '11)*, pp. 1–6, July 2011.

[51] V. Singh, J. Ott, and I. D. D. Curcio, "Predictive buffering for streaming video in 3G networks," in *Proceedings of the 13th IEEE International Symposium on a World of Wireless, Mobile and Multimedia Networks (WoWMoM '12)*, June 2012.

[52] S. Nazir, Z. Hossain, R. Secchi, M. Broadbent, A. Petlund, and G. Fairhurst, "Performance evaluation of congestion window validation for DASH transport," in *Proceedings of the Network and Operating System Support on Digital Audio and Video Workshop (NOSSDAV '14)*, p. 67, ACM, New York, NY, USA, 2013.