IEEE Access
Multidisciplinary ⁝ Rapid Review ⁝ Open Access Journal

Received December 14, 2021, accepted January 11, 2022, date of publication January 20, 2022, date of current version January 27, 2022.

Digital Object Identifier 10.1109/ACCESS.2022.3145185

Network Capacity Estimators Predicting QoE in HTTP Adaptive Streaming

SANNA LAINE AND ISMO HAKALA, (Member, IEEE)
Kokkola Unieverisity Consortium Chydenius, University of Jyväskylä, 67701 Kokkola, Finland
Corresponding author: Sanna Laine (sanna.m.laine@jyu.fi)

ABSTRACT The aim of adaptive HTTP streaming technology is preserving the best possible video streaming quality for viewers in heterogeneous network conditions. This can be achieved by making multiple quality versions of the video available. Switching between versions during playback should be imperceptible and fluent. The decision about quality-level switching is typically based on network capacity estimation and buffer occupancy, which predict the risk of stalling. Since quality-level switching and stalling are directly evident to the user, they are often classified as influence factors of quality of experience (QoE). In this paper, we observe different network capacity estimators and buffer behavior in limited network conditions and study how the estimators predict QoE. The challenges of variable bitrate (VBR)-encoded video are considered. We also propose two new estimators to predict QoE. One compares segment fetch time to segment playback time, while the other explores the difference of throughput and average download rate. As segment duration may influence HTTP adaptive streaming (HAS) playback in unstable conditions, the findings are tested with four segment lengths. Moreover, streaming quality is analyzed in a testbed using two popular web players to reveal possible effects of the players' features.

INDEX TERMS Adaptive algorithm, HTTP adaptive streaming, network capacity estimators, player performance, streaming media, quality of service, quality of experience.

I. INTRODUCTION

The high global Internet penetration rate has enabled the massive growth of streaming video on demand (SVoD) services. Cisco [1] has predicted that Internet video usage will reach 82 percent of global Internet traffic by 2022. The main actors in the video delivery chain are content providers, content delivery network (CDN) operators, Internet service providers (ISPs), and application designers [2]. Although these parties have their own criteria for developing services, the quality of experience (QoE) of the end user is their common interest for customer satisfaction. ITU-T FG IPTV [3] defines QoE as "the overall acceptability of an application or service, as perceived subjectively by the end user". QoE is subjective and depends on a user's experiences and context. QoE data can be collected from test environments involving humans. There are also standardized models like ITU-T P.1203 recommendation for assessing the QoE of HAS.

Before delivering a video, the content provider makes decisions about encoding and compression that affect video quality. The aim of on-demand streaming is to transmit and display the stored video with the best quality possible from the user point of view. The transmission channel has characteristics like the available bandwidth, packet loss, delay, and jitter, which can affect QoE. Video transmission in the network is controlled by the rules defined by streaming and transmission protocols. Since the 2010s, HAS technology has overtaken the streaming protocol field. In HAS, all requests are done via HTTP on port 80, similar to plain web browsing. Thus, the streaming traffic is capable of traversing firewalls and proxy servers. HAS does not need a persistent connection between the server and player, and it can utilize existing content delivery networks. No special streaming servers are needed. In practice, the four following HAS technologies currently share the market: Apple's HTTP Live Streaming (HLS), Microsoft Smooth Streaming (MSS), Adobe's HTTP Dynamic Streaming (HDS), and MPEG's Dynamic Adaptive Streaming over HTTP (DASH or MPEG-DASH). DASH is the first HTTP-based adaptive bitrate streaming solution that is an international standard. In this paper, HLS is used, since it is the most widely-used streaming protocol.

In HAS technology, the video is encoded into multiple quality versions. Furthermore, each quality version is divided into sections of few seconds called segments. Using HTTP

The associate editor coordinating the review of this manuscript and approving it for publication was Nishant Unnikrishnan.

GET messages, the client requests each segment separately. Segmentation reduces network wastage compared with traditional progressive downloads, where the entire video is downloaded with a single request. Switching the quality level is possible in segment boundaries because quality levels are segmented evenly. If the streaming conditions change, the client can request the next segment in a different quality than used for the previous one. While downloading segments, the client collects and monitors the data needed to evaluate the streaming conditions. Collected data can include the observed throughput for each segment, buffer occupancy information, and possibly, the central processing unit (CPU) load. By combining and analyzing the collected data, the network conditions and system capacity are assessed. The goal is detecting the limits where the current bitrate should be switched higher or lower to better match prevailing streaming conditions. The choice between bitrate options is based on estimated streaming conditions and adaptation logic. The adaptation logic includes the rules that define how conservatively or aggressively the bitrate is changed. In addition to the adaptation logic, buffering strategies and segment duration may affect the streaming quality in fluctuating network conditions. Buffering strategies help optimize playback performance by trying to prevent stalling, but at the same time, enabling fast startup and minimizing data wastage.

A typical segment duration is between 2 and 10 seconds. Apple recommends the target duration of a segment to be 6 seconds [4]. Short segments enable faster reactions to changing streaming conditions because the segment duration defines the bitrate switching interval. Every segment must start with an I-frame, and this lowers the encoding efficiency compared with the same bitrate videos with fewer I-frames. Requesting each segment separately causes a higher overhead compared with traditional streaming protocols. While the overhead can be decreased by lengthening the segment duration, longer segments may increase the initial delay.

Like other streaming techniques, HAS is vulnerable to network interference. Various factors affect delivery efficiency in packet-switched networks; such factors include the available bandwidth; network congestion; bit errors; capacity restrictions on the client; data processing on the media server, routers or switches; and interference in the transmission medium. HAS faces extra challenges because it uses TCP as the transmission protocol. TCP performs best with a steady stream of data packets. Thus, the sequential HTTP requests, creating an on-off pattern, present a challenge for TCP. The relationship of HAS and TCP performance is considered, for example, in works by Hu *et al.* [5] and Huang *et al.* [6].

This study explores the relationship between selected quality of service (QoS) estimators and the QoE influence factors of HAS playback. In HAS, the most common QoE influence factors at application-level are a long initial buffering time, interruptions due to rebuffering, and decreased quality of the segments (and switching between them) [7]. Delay in the startup phase and rebuffering are shown directly to the user, and HAS tries to minimize them by decreasing the segment quality. Hence, quality changes —even noticeable ones— may be unavoidable; however, too-frequent quality switching or even distracting bitrate oscillation can be eliminated with a decent adaptation algorithm. Garcia *et al.* [8] also detected that changes between high-quality videos are less noticeable than changes between low-quality ones. They suggested preparing more video versions at low quality to make the quality change gradual and less noticeable to the user. Typically, adaptation methods use network capacity estimation and/or buffer occupancy information to choose a suitable bitrate. Even many of the model based approaches are developed around these QoS metrics like a DASH rate adaptation algorithm QUETRA, developed by Yadav *et al.* [9].

As network capacity and buffer occupancy change before the user observes an improvement or reduction in quality, they predict the QoE influence factors of the playback. Often, predictive metrics are modified to be more applicable in adaptation algorithms. In this study, we monitor these metrics under changing network conditions to reveal their features and possible weaknesses in bitrate adaptation. To cause changes in HLS streaming quality, a variable bitrate (VBR)-encoded test video was streamed in altered network conditions by varying the available bandwidth and inducing packet loss. Two popular video players designed for web playback were chosen for analysis.

This study makes the following contributions in the HLS streaming context:

- It shows effects of network impairments on bandwidth estimators and buffer occupancy (i.e. on typical metrics used in adaptation algorithms);
- It examines how well common bandwidth estimators, as well as our two proposed estimators, and buffer occupancy can predict QoE influence factors;.
- It examines effects of lengthening the segment duration on QoE influence factors in unstable streaming conditions; and
- It uses two players in our test environment to examine the possible effect of players' features.

The paper is organized as follows: Section 2 discusses the related work, and Section 3 introduces streaming condition estimators used in this paper. Section 4 presents our testbed, while Section 5 describes the HLS streaming behavior in the test environment, where network conditions are altered. The effect of segment duration is also examined. Section 6 discusses the findings in this study, and Section 7 concludes the paper.

II. RELATED WORK

The performance of media players depends largely on rate adaptation methods. In addition to throughput-, buffer-, and hybrid-based approaches, Yadav *et al.* [9] divide adaptation approaches into QoE-centric, queuing model, and non-normative based. The simplest throuhput-based methods evaluate future throughput with the most recently arrived segment. Buffer-based methods use different thresholds to prevent buffer underflow or overflow. When throughput metrics and buffer occupancy are used together for assessing streaming conditions, the bitrate version can be selected by assessing the TCP throughput first and then fine tuning the selection based on the buffered media time. Karn *et al.* [10] developed an algorithm that predicts throughput but makes the final decision about level switching based on buffer occupancy. Particularly, when the buffer occupancy is between two threshold values, the algorithm will keep the current quality level regardless of throughput estimate. That is to avoid unnecessary quality switching when there are two competing clients. Tian and Liu [11] generated a rate adaptation algorithm that uses TCP throughput estimation in addition to an adjustment factor, which is the product of the buffer size adjustment, buffer trend adjustment, and video segment size adjustment functions. QoE-centric based methods consider throughput and buffer occupancy in order to avoid radical or frequent quality level changes, and other factors that are known to influence on QoE. In queuing model based approaches, HAS client is a queuing system, where queue length is the buffer occupancy. QUETRA [9] selects the quality level of the segment so that buffer occupancy converges to the ideal value in the estimated network throughput. Thus, also QUETRA combines buffer-based and throughput-based approaches. Into non-normative approach Yadav *et al.* [9] categorize methods that use less general goals or means when selecting quality level. These include for example server-side quality selection aiming for fairness among clients. To conclude, even more sophisticated adaptation approaches use network throughput estimation or buffer occupancy monitoring.

VBR videos bring more challenges in bitrate selection. The more a VBR video's bitrate varies from the target encoding bitrate, the more leeway is needed to prevent buffer underflows. It is not uncommon for videos on the Internet even to double the advertised bitrate occasionally. In addition to buffer size, various smoothing techniques are applied to improve playback quality in streaming VBR videos. Le *et al.* [12] used video bitrate estimation with a moving average to evaluate the capacity sufficiency more precisely for VBR videos. The adaptation algorithm of Dubin *et al.* [13] estimated the median bandwidth instead of average based on previous segments to obtain more stable estimation. These researchers also suggested that playlist files should include each segment rate in addition to the average bitrate of the entire quality level.

The effects of segment duration have been considered in various studies. Sideris *et al.* [14] observed in their experiment that a longer segment duration achieved a better QoE level. They deduced that downloading shorter segments prohibits the TCP's sending window from reaching high values, which causes the adaptation logic to remain at lower quality levels. Islam and Khan [15] observed that downloading one large segment is faster than downloading multiple smaller segments; they considered the option that, instead of switching to a lower bitrate version, the segment duration could be varied in insufficient network conditions. In addition, Liu *et al.* [16] studied the possibility of using segment duration in the rate adaptation. They developed a rate adaptation method, that estimates the minimum segment duration for producing a smoothed HTTP/TCP rate, representing the current network capacity.

Nguyen *et al.* [17] compared streaming with fixed segment lengths of 2, 5, and 10 seconds in networks with different round trip times (RTTs) and using instant and smoothed capacity estimation methods. They found that advantages of longer segments arise when the RTT is increased. Videos delivered with longer segments reach a higher average bitrate during streaming, especially with the instant throughput–based adaptation method. They also discovered that using a shorter segment duration reduces the occurrence frequency of buffer underflows. In another study, Mondal *et al.* [18] explored YouTube's bitrate and quality adaptation algorithm. They found that YouTube uses a parallel downloading of segments and segment length changing to offer the best possible quality with minimum data wastage.

In our research, we bring together and compare some common estimators, introduced in Section III, and evaluate how QoE influence factors can be predicted with them. These estimators can further be used in adaptation algorithms. By modifying the introduced metrics, we form two new streaming quality estimators. The analysis is performed using a VBR-encoded video. Although HTML5 tags enable embedding videos directly in a webpage, adding adaptive bitrate streaming, live streaming, and other functionality requires using HTML5 Media API and JavaScript. For that reason, a readymade HTML5 player is often the most straightforward solution. Unlike in the papers mentioned previously, in this study, two common readymade web players are chosen as test players.

III. NETWORK CAPACITY AND BUFFER OCCUPANCY ESTIMATION

The traditional method for determining the most suitable bitrate version is assessing the network capacity and/or monitoring the client buffer occupancy. To avoid reacting to short-term throughput variation caused by TCP congestion control, a smoothed throughput estimation can be used to detect more persistent bandwidth changes [19]. The simplest method for assessing the network capacity T is measuring the segment fetch time (SFT) and dividing the segment size l_{size} by it, that is,

$$T(i) = \frac{l_{size}(i)}{SFT_i}.$$

Here, SFT_i denotes a period of time from the time instant t_i of sending a GET request for a ith media segment to the instant of receiving the last bit of the requested media segment, that is, the time consumed downloading a segment of size $l_{size}(i)$. The network capacity for the next segment request interval $i+1$ can be estimated as

$$T'_e(i+1) = T(i). \tag{1}$$

The longer the media segment duration, the smoother the throughput estimation is in equation (1). Liu *et al.* [19] estimated segment size l_{size} by multiplying segment duration l_{dur} with segment bitrate l_{br}, obtained from the playlist file. Hence,

$$T_e'(i+1) = l_{br}(i) \cdot \frac{l_{dur}}{SFT_i} = l_{br}(i) \cdot \mu. \tag{2}$$

Method (1) takes only the previous segment fetch time into account when estimating the network throughput for the next segment interval. To smooth the estimation more, the exponentially weighted moving average (EWMA) of segment throughput

$$T_s(i) = \begin{cases} (1-\delta)T_s(i-1) + \delta T(i), & i > 1 \\ T(i), & i = 1 \end{cases}$$

can be used. Parameter $\delta \in]0,1[$ is a weighting value (smoothing factor). Following [20], the smoothed throughput estimate for the download interval $i+1$ can now be formulated as

$$T_e(i+1) = T_s(i). \tag{3}$$

A smoothed bandwidth may cause a late reaction to a large throughput decrease. In these situations, the buffer should be big enough to prevent stalling.

In HAS streaming, the data are transferred periodically. A segment fetch period, the time between two consecutive GET requests, may include long idle periods. Thus, the average download rate can be much lower compared with the throughput. The average download speed in a segment fetch period is

$$A(i) = \frac{l_{size}(i)}{t_{i+1} - t_i}. \tag{4}$$

This represents the average speed at which the client can receive a segment due to restrictions set by the throughput or playback buffer. Here, t_i is the timepoint of the GET request for the ith segment. Akhshabi *et al.* [21] used EWMA-smoothing on the average download speed in a constant, 2-second period for exploring adaptation algorithms. In that case, $A'(i) = m_i/(2 \text{ seconds})$, where m_i represents all media data downloaded in the ith 2-second period. In the following, we use EWMA smoothing on equation (4) for estimating the average download rate for the next segment interval, that is

$$A_e(i+1) = A_s(i),$$

where

$$A_s(i) = \begin{cases} (1-\delta)A_s(i-1) + \delta A(i), & i > 1 \\ A(i), & i = 1. \end{cases}$$

In this paper, A_e is applied to form a new estimator. The difference $\Delta(i) = T(i) - A(i)$ gives information about the bandwidth utilization. When the playback is in a steady state, that is, the playback buffer is full, $A(i)$ follows the average video bitrate, and the difference $\Delta(i)$ increases. The less time a player uses on idle periods, the closer $A(i)$ becomes to $T(i)$; that is $\Delta(i)$ decreases. Decreasing the difference means that the client strives to maximize the use of the bandwidth as the player is in the buffering state. In addition to operating in the startup phase, the buffering state is on every time the buffer occupancy decreases due to insufficient streaming conditions for the current video bitrate, that is, when media data are removed from the playback buffer faster than they are received. Due to these features, the EWMA-smoothed difference,

$$\Delta_e(i) = T_e(i) - A_e(i), \tag{5}$$

is the other of our two proposed new streaming quality estimators. One option for estimating the threshold value for Δ_e is monitoring the initial buffer filling phase. When the buffer is filled for the first time, the client uses the maximum capacity and Δ_e is its smallest. The closer to zero the difference becomes during the filling phase, the more efficiently the client can utilize the available bandwidth. If Δ_e later approaches the value observed during initial buffer filling, the playback is closer to transitioning into the buffering state.

If the video bitrate occasionally shows high variation, even inside the same HAS quality level, it may not be enough to compare only the throughput to the bitrate for the adaptation methods. To fade out absolute throughput measuring, the approach to observing the relationship of the segment fetch time and segment playback time was examined, that is,

$$S(i) = SFT_i / l_{dur}.$$

If $S(i)$ exceeds a threshold value, $\lambda = 1$, segment i is received slower than one is played out. This results the buffer occupancy to fall. Vice versa, the value of $S(i)$ under 1 depicts that the current throughput is sufficient for the video and the buffer occupancy can grow if it is not yet full. In practice, a value of λ below 1 should be selected to give more time, for example, for decoding. Metric $S(i)$ evaluates the buffer occupancy development direction. It is not independent of $A(i)$ and $T(i)$, as the following equation shows:

$$S(i) = \frac{SFT_i}{l_{dur}} = \frac{t_{i+1} - t_i}{l_{dur}} \cdot \frac{A(i)}{T(i)}.$$

To form a streaming quality estimator from $S(i)$, we used an EWMA-smoothed version of it:

$$S_e(i+1) = S_s(i), \tag{6}$$

where

$$S_s(i) = \begin{cases} (1-\delta)S_s(i-1) + \delta S(i), & i > 1 \\ S(i), & i = 1. \end{cases}$$

While monitoring S_e, buffer occupancy can be estimated by comparing the received media time to the passed time from the initial buffer filling. Basically, all information needed comprises timestamps of sent GET requests from the network level. If application-level information of the video timeline position value is available, an accurate buffer occupancy can be deduced. As the segment duration l_{dur} is constant, and the player requests a segment only after all the previous ones are received, the buffer occupancy b at timepoint t is

$$b(t) = (i-1) \cdot l_{dur} - p(t), \quad t \in [h_i, h_{i+1}[. \qquad (7)$$

Herein, h_i is the time when the ith segment has arrived and $p(t)$ is the video timeline position value at time t. Using $p(t)$ gives more accurate estimation than using real time in the buffer evaluation in pursuance of S_e monitoring. Particularly, method (7) takes stalling occurrences into account. As segments are moved into the playout buffer as a whole, $b(t)$ forms a sawtooth line. In the buffer occupancy figures given below, we depict the buffer occupancy only in the GET request points $b(t_i)$, in which case, the buffer always contains at least one segment.

IV. THE TEST ENVIRONMENT

During the tests, we used two HTML5 video players that are built for web playback—the commercial JW Player [22] and open source Video.js player [23]. Both are commonly used for professional online video deployments. Adaptive streaming is possible with HLS, MPEG-DASH, and RTMP protocols. In our tests, HLS streaming and the current market-leading web browser, Google Chrome, were used. Video was streamed from a normal web server with the Windows Server 2008 R2 operating system, which uses the Compound TCP (CTCP) version [24]. The media data were sent on the network layer in 1,500 byte–sized packets. CTCP uses the delay- and loss-based congestion avoidance approaches. We also observed that the client used delayed acknowledgment, informing only every other data packet.

Changing network conditions were generated during the streaming tests with the Linktropy 5500 WAN emulator [25] by adjusting the available bandwidth and packet loss rate. During the playbacks, the media data traffic was monitored with the Wireshark network analysis tool [26]. The playback progress information was tracked with JavaScript, utilizing the players' API methods. The playback progress information was combined with timestamps of the GET requests recorded by Wireshark to estimate the buffer occupancy at the GET requests using equation (7). The test environment proved to be isolated enough to provide very little variation in the metrics monitored during playbacks with the same settings. The test setup is illustrated in Fig. 1.

As it is commonly applied in video quality research papers, the VBR encoded Big Buck Bunny [27] animation was chosen as a test video. To assure some results, also another test video, Elephants Dream [28], was used. The videos were re-encoded suitable for HLS transmission. The specifications of videos are shown in the Table 1. The client machine information is in the Table 2. Fig. 2 presents the bitrate profiles of test videos with 2-second long segments. Each video and audio frame has a display timestamp that defines when the frame should be rendered. The figure shows the size of frames in each display second. The blue dashed line depicts the overall average bitrate of the video. The blue solid line is the

TABLE 1. Test video information.

	Test video 1	Test video 2
Name	Big Buck Bunny	Elephants Dream
Duration	10min 34s	10min 53s
Frame rate	60 fps	24 fps
Target bitrate	5 Mbps	2 Mbps
Resolution	1080p	720p
Codec	AVC+AAC	AVC+AAC

TABLE 2. Client machine information.

Client machine	
Model	Dell Latitude 7390
Processor	Intel Core i5
Number of CPU cores	4
CPU Cache	6MB
CPU Base Frequency	1.7GHz
GPU	UHD Graphics 620
Display	1920 x 1080 x 60 Hz, 32 bits/pixel
Operating system	Windows 10

20-second moving average of the bitrate profile. The content dictates the variation in the bitrate profile.

V. RESULTS

In this section, the effects of different bandwidth conditions and packet loss rates on network capacity estimators and the buffer behavior are observed. For HDS, the client's playback buffer size is roughly recommended to be at least three times the segment duration [29]. The specification states that the buffer length should provide minimal playback disruptions while considering factors like network conditions, desired latency, desired start times, and effects on server scalability. The DASH and HLS specifications do not stipulate the buffer length. In our test setup, the maximum buffer size in the media time was 25s for JW Player. The default buffer size of Video.js is 60s, but it was also set to 25s in the test environment. As a default, both players gathered only 1 or 2 segments to the buffer before starting the playback, when the segment length was 2 seconds. With longer segments tested, players started the playback after receiving one segment. This led to short initial delays (even at the expense of smooth playback—a small initial buffer occupancy may cause stalling occurrences right at the beginning of the playback).

A. BANDWIDTH LIMITATION

For years, service providers have relied on network over-provisioning as a solution to traffic fluctuations. Reserving more bandwidth, than the expected traffic load, provides readiness to serve future customers, although it is not energy efficient. However, over-provisioning does not solve all situations in a network. Not all routers prioritize real-time applications, and UDP-based flows lacking congestion control may flood the network [30]. When multiple flows compete for their fair share of the link, the throughput decreases. Especially, if the link is shared between other adaptive streaming flows with a temporal overlap of the on-off periods, the fair share may be estimated incorrectly. This

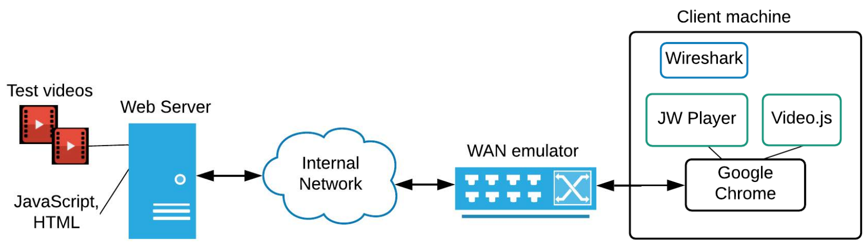
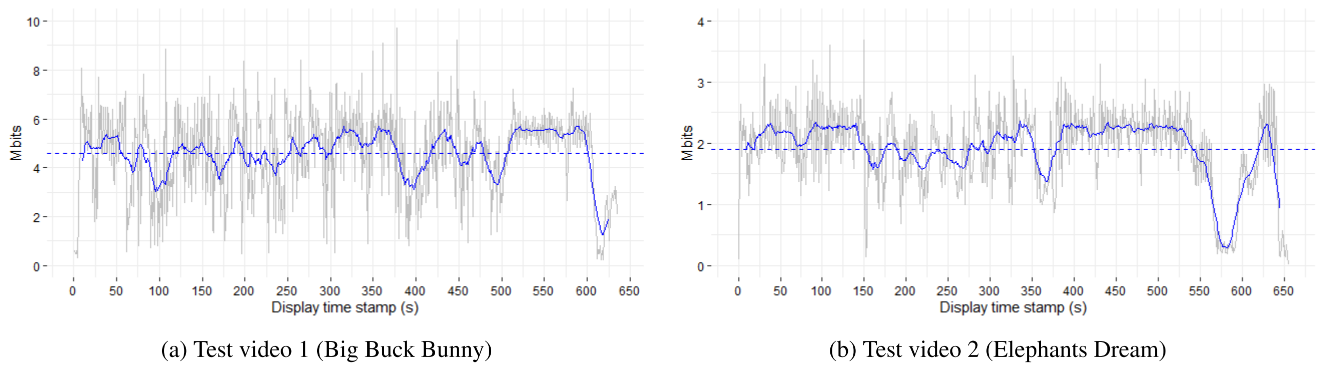


FIGURE 1. Test environment. Video segments are fetched from a web server and transported through a campus network. Before delivering packets to the client, streaming conditions are altered with a physical WAN emulator. Network traffic is monitored with the Wireshark packet capture program.



(a) Test video 1 (Big Buck Bunny)

(b) Test video 2 (Elephants Dream)

FIGURE 2. Encoding bitrate profiles with moving averages of Test video 1 a) and Test video 2 b) in bits/display time.

causes instability in video quality, unfairness, and bandwidth underutilization [31].

Larger video player buffer sizes, better playing strategies, and improvements in TCP have decreased network throughput requirements [32]. For constant bitrate videos, Biernacki and Tutschku [32] assessed that the network throughput should exceed the video bitrate for about 15 percent for smooth transmission. In a simplest case, an adaptation algorithm chooses the maximum of the bitrates that meet the condition $\gamma \cdot l_{br} \leq T_e$, where $\gamma = 1.15$ is a coefficient that is evaluated to guarantee enough bandwidth to overhead traffic. With VBR videos, a throughput exceeding the average bitrate by 15 percent may not be enough.

To examine the estimators' behavior and players' performance, we regulated the available bandwidth with the WAN emulator using the two players. In a test case shown in Fig. 3, the bandwidth is first set to be 1.7 times the average bitrate, then reduced to 1.2 times, changed to 1 time, and increased back to 1.2 times the average bitrate of test video 1. This kind of sudden bandwidth change may be caused by other clients connecting to share the link. Figure shows the received media data per second (gray line) and estimators T_e (green line), A_e (blue line), and Δ_e (brown line) for both players. The initial value of the smoothing factor δ is chosen to be 0.2, following Akhshabi *et al.*'s [21] article. The time is set to start from the timestamp of the GET request for the first segment. During the tests, a 2-second-long segment length was used.

In the first bandwidth period ($bw = 1.7 \cdot average$ ($video\ bitrate$)), players fill their playback buffers as fast as possible. This causes Δ_e to decrease to near zero as all the available bandwidth is utilized. After that, A_e and T_e diverge from each other, and the variation of A_e increases. An increasing Δ_e value denotes that the network connection will allow higher bandwidth utilization than is used in video streaming. At its highest, Δ_e rises to 4.4 Mbps; that is, at this point, the client is using over 4 Mbps less bandwidth than offered. The decrease of A_e results from the idle periods in the data transmission. However, T_e stays up since packets are received as quickly as they were earlier, although requested less frequently. The playback is in a steady state; that is, the buffer is full and the next segments are requested less frequently to prevent buffer overflow.

In the second period, the available bandwidth is decreased to 1.2 times the average video bitrate. Estimates A_e and T_e can still keep their distance from each other, meaning that clients can have idle periods and playback is not vulnerable to stalling. In the third and fourth periods, Δ_e approaches zero. Both A_e and T_e are near the emulated bandwidth; that is, the players request each segment as soon as they have received the previous one. This may indicate decreasing buffer occupancy. The situation is not much improved when the available bandwidth is raised back to the second period level ($bw = 1.2 \cdot average$ ($video\ bitrate$)). The behavior of the estimators is extremely similar in both players.

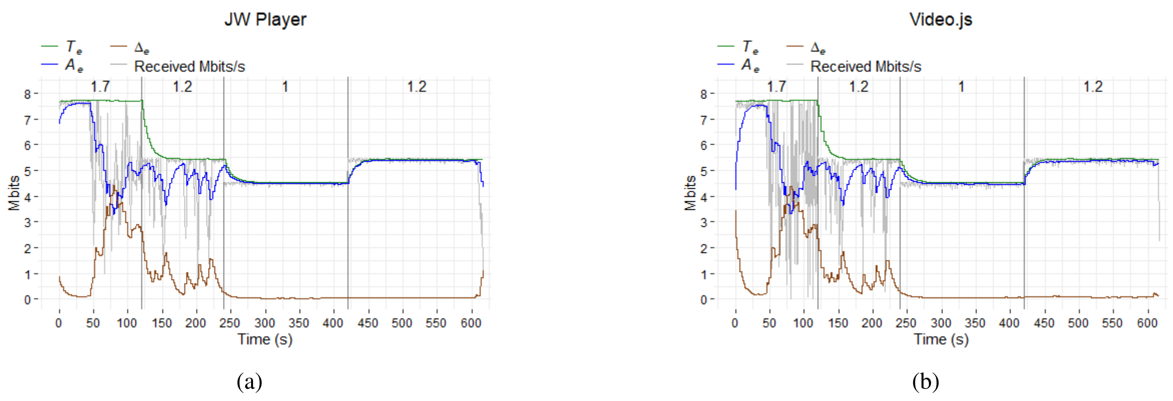


FIGURE 3. Received Mbits/s, and estimators T_e, A_e, and Δ_e (smoothing factor $\delta = 0.2$) in changing bandwidth conditions with JW Player a) and Video.js player b). The segment duration is 2s, and the maximum playback buffer size, in media time, is 25s.

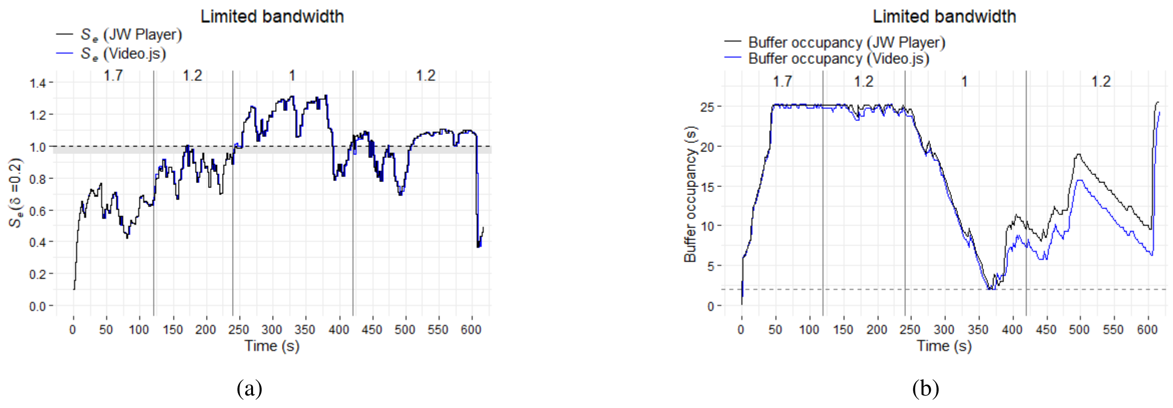


FIGURE 4. Smoothed estimator S_e ($\delta = 0.2$) a) and buffer occupancy b) for JW Player and Video.js in variable bandwidth conditions. The segment duration is 2s, and the maximum playback buffer size, in media time, is 25s.

In Fig. 4, the new estimator S_e is applied, and the buffer occupancy for both players during the same test runs as above is depicted in time instances of GET requests using equation (7). In the S_e chart, the threshold value is $\lambda = 1$ and smoothing factor is $\delta = 0.2$. In Fig. 4a, S_e stays below the threshold value λ in the first two periods, indicating sufficient streaming conditions with both players. In the first period, the maximum of S_e is about 0.75; that is, downloading a segment takes roughly less than 75 percent of the time it takes to play it back. This is enough for building up the buffer occupancy. Both players can also hold the buffer fullness well in the second bandwidth period.

In the third period, S_e rises above the threshold value. On average, $S_e \approx 1.14$ for both players in the third period. This means that the streaming would need approximately 14 percent more bandwidth than offered with the current bitrate. The time since the arrival of the first segment reaches a received media time of about 360 seconds with JW Player. Thus, it can be assumed that all the received media time is played out, and buffer underrun will take place. The buffer decrease is proved by Fig. 4b, where a more accurate buffer approximation, with the video timeline position, is used. The user will not see any changes in playback quality until the buffer has drained or reached the threshold defined by the player, causing stalling. Both playbacks stall twice in the third period. The last test period looks extremely different from the second test interval, although the available bandwidth is the same in both periods. This is explained by the variable bitrate of the test video. The last 2 minutes of the content requires a higher bandwidth than the first part (see Fig. 2a). S_e exceeds the threshold value and stays above it for over a minute.

QoE influence factors (total stalling times and occurrences), were observed in conditions where the available bandwidth was not changed in the middle of the playback. The test video was played five times on six bandwidth levels with both players. Fig. 5 shows, that when the available bandwidth decreases below $1.2 \cdot \textit{average}$ (*video bitrate*), stalling starts to appear; that is, QoE starts to decrease. Thus, the threshold limit for estimator T_e should be 20 percent larger than the average video bitrate for this video; in other words, selecting segments in such a way that $T_e > 1.2 \cdot l_{br}$ should prevent stalling. Our previous observations with estimators Δ_e and S_e and the buffer occupancy values suggest switching quality level when the bandwidth drops under $1.2 \cdot \textit{average}$ (*video bitrate*) because of the bitrate variation of the test video. This result is in line with Fig. 5. In the setup above,

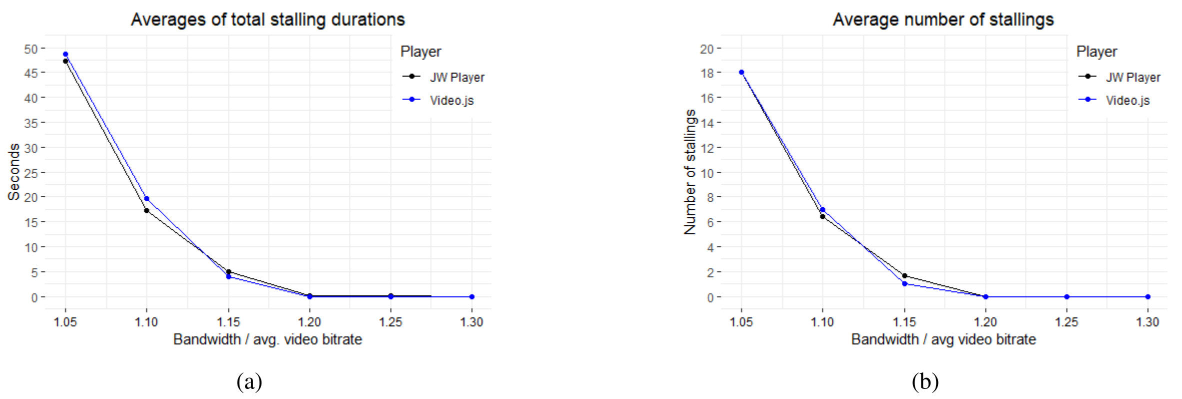


FIGURE 5. Averages of total stalling durations a) and times b) in constant relative bandwidth values with the test video. The playback was done five times with each bandwidth value (1.05–1.30 times average video bitrate). The segment duration is 2s, and the maximum playback buffer size, in media time, is 25s.

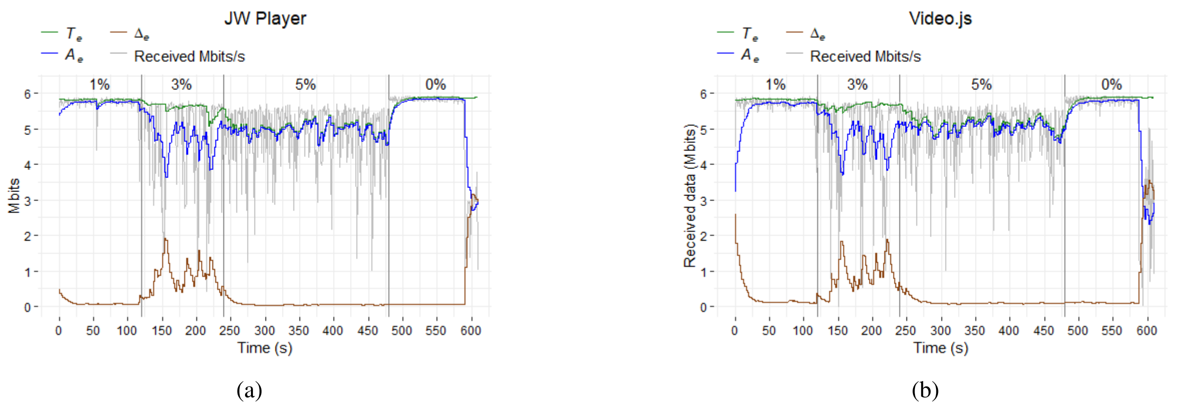


FIGURE 6. Received Mbits/s, and estimates T_e, A_e, and Δ_e (smoothing factor $\delta = 0.2$) in changing packet loss conditions with JW Player a) and Video.js player b). The segment duration is 2s, and the maximum playback buffer size, in media time, is 25s.

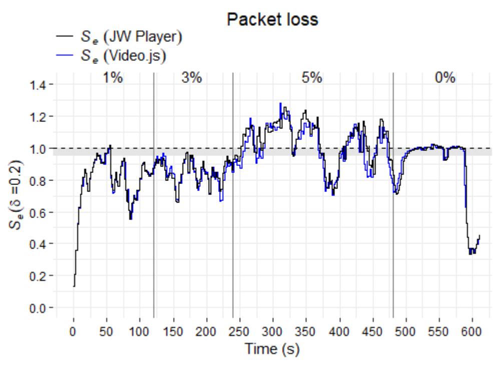
all the estimators concerned work fairly reliably. However, the constant bandwidth does not usually compare with the reality. In the next section, packet loss is added to the channel to cause more bandwidth fluctuation.

B. PACKET LOSS

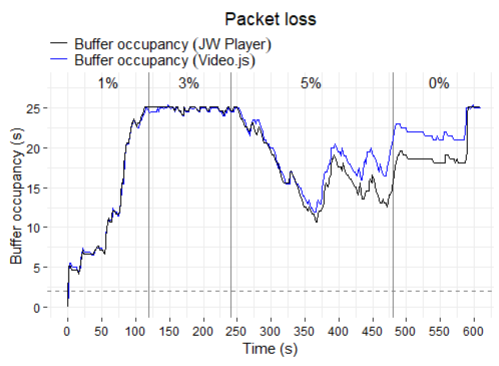
The most common cause of packet loss in wired networks is congestion. Another cause of losses is transmission errors resulting in corrupted packets, which are then rejected. Device-based reasons include the performance of routers or switches that are unable to handle all traversing traffic or damaged cables. Wireless networks are more vulnerable to packet loss as the signal strength weakens due to multipath fading. The popularity of mobile devices makes packet loss common. As HAS protocols run on top of TCP, all lost or corrupted packets are resent. If the available bandwidth is high enough, the increased retransmissions may not manifest to the user. Many retransmissions, and especially TCP retransmission timeouts, can cause delay, throughput fluctuation, and finally, buffer underflow. Video image artifacts, such as blockiness or blurring, are not typical in TCP streaming.

The influence of packet loss was tested while keeping the available bandwidth constant. From previous tests, we concluded that bandwidth exceeding the encoding bitrate by 20 percent is just enough for playing the video back flawlessly, but this may cause the buffer occupancy to decrease in the final part of the video. To ensure that insufficient throughput is caused by the packet loss, the available bandwidth was set to $1.3 \cdot average\ (video\ bitrate)$ in the test setup. The loss rate was set first to 1%, then increased to 3%, and then increased again to 5%. The final part was played without packet loss to see how quickly estimators reacted to improved network conditions. The WAN emulator discards packets randomly based on the specified packet loss rate. Dropped frames also consume link bandwidth.

The received media data per second and estimators T_e, A_e and Δ_e are shown in Fig. 6 for both players. Packet loss affects the throughput and causes variation in estimators. Although δ is chosen as a way to smooth out T_e, the variation may still lead to failure in selecting optimal bitrate, when using T_e alone. Estimators T_e and A_e follow each other when the player tries to fill the buffer, that is, Δ_e approaches zero with only a slight variation. In the second period (3% loss), Δ_e rises, indicating that the buffer fills up momentarily. When the packet loss is removed in the final period, T_e and A_e start to rise again near the emulated bandwidth level. It takes almost

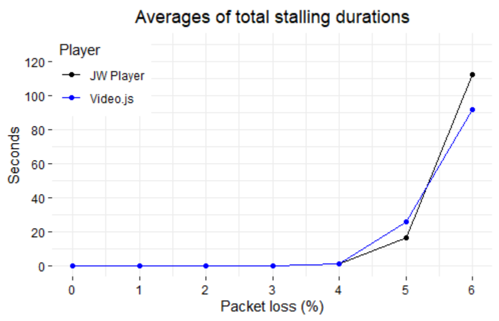


(a)

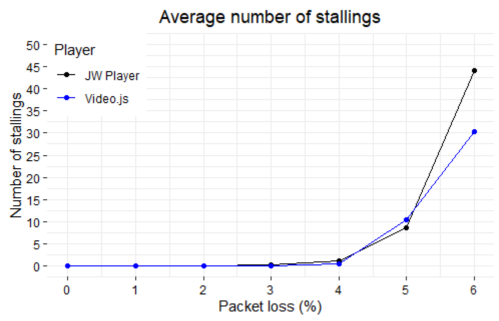


(b)

FIGURE 7. **Smoothed estimator S_e ($\delta = 0.2$) a) and buffer occupancy b) for JW Player and Video.js at four different packet loss levels. The segment duration is 2s, and the maximum playback buffer size, in media time, is 25s.**



(a)



(b)

FIGURE 8. **Averages of total stalling durations a) and times b) in different packet loss conditions with the test video. The playback was done five times in each bandwidth value. The segment duration is 2s, and the maximum playback buffer size, in media time, is 25s.**

a minute for T_e to reach its maximum, but most of the rise takes place in a few seconds.

Fig. 7 depicts the behavior of estimator S_e and the buffer occupancy during previous test runs. In 1% and 3% packet losses, S_e stays below the threshold value, indicating that the channel can deliver packets in sufficient speed to render video. In a 5% packet loss period, S_e rises above the threshold value of λ. In this period, the average of S_e is 1.04 with JW Player and 1.01 with Video.js. Stalling occurrences are still avoided. The buffer occupancy is, at the lowest, about 11s with JW Player and 12s with Video.js in the third period.

Packet loss naturally slows down the buffer filling and causes the initial delay to lengthen. However, neither players gather more than 1–2 segments to buffer before starting the playback. This already causes difficulties in the beginning as both players can raise their buffer level only to about 10 seconds during the first minute. As could be assumed based on estimator Δ_e, the players manage to fill in their playback buffer in the second packet loss period. From the behavior of estimator S_e, it could be deduced that, in the 5% packet loss period, with the used bandwidth, the level of media data decreased in both players' playback buffer.

Fig. 8 shows the averages of total stalling times and stalling occurrences, when the packet loss ratio is kept constant during the whole playback time and the available bandwidth is limited to $1.3 \cdot \textit{average (video bitrate)}$. The video is played back five times with each packet loss ratio. The stalling time starts to increase when the loss ratio exceeds 4%.

Along with the packet loss, the throughput conditions were more realistic during the tests in this section. Naturally, the fluctuating throughput affected the behavior of T_e the most. Estimator Δ_e behaved similarly to the case with constant available bandwidth. Estimator S_e varied a bit more than it did without packet loss, but as it only indicates the incoming data in relation to played data, it is easy to interpret.

C. SEGMENT LENGTH

Segment length is usually decided on the server side. The decision depends on the terminal device and video content. Short segments enable quick adaptation to changing network conditions. Every segment starts with a key frame; thus, long segments allow higher efficiency in encoding. In addition, fewer requests are needed, which reduces the overhead. For the HLS protocol, a segment length of 6 to 10s is often recommended. However, for example, Bitmovin [33], suggested HLS segment sizes of around 2 to 4s to achieve a good compromise between encoding efficiency and flexibility for stream adaptation to bandwidth changes. This section

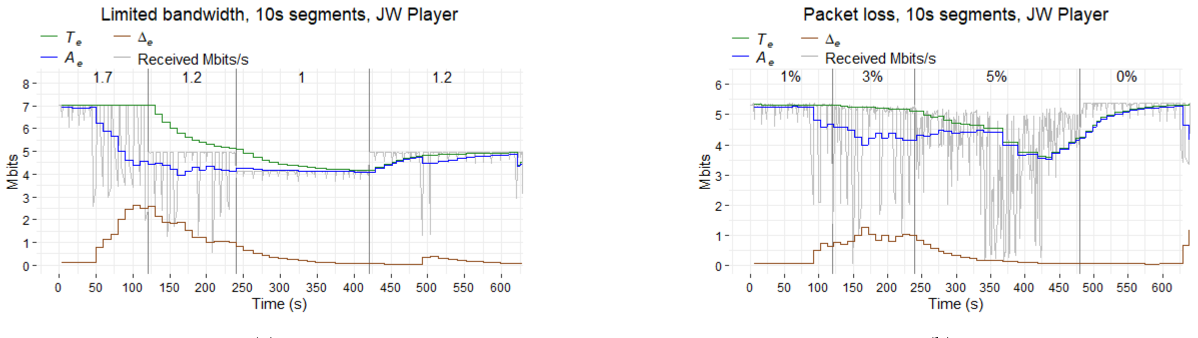


FIGURE 9. Received media data in Mbits/s and estimates T_e, A_e, and Δ_e ($\delta = 0.2$) for JW Player in variable, limited bandwidth a) and packet loss conditions b). A 10-second segment video is employed with a maximum playback buffer size of 25s.

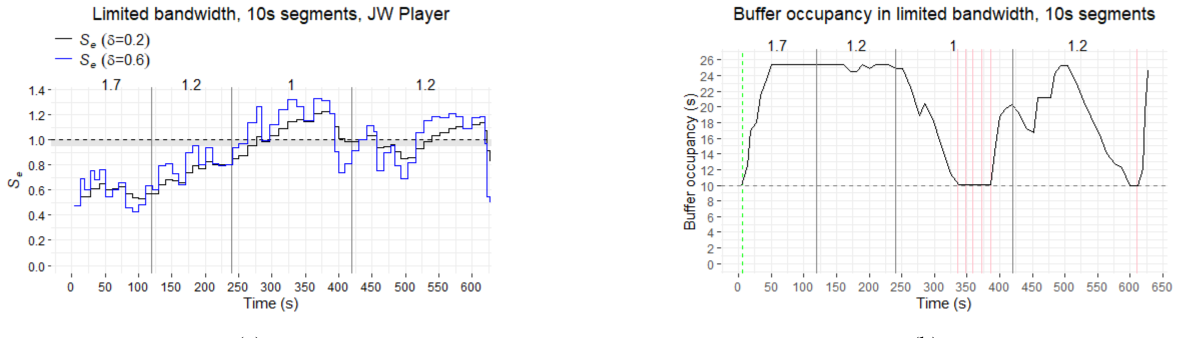


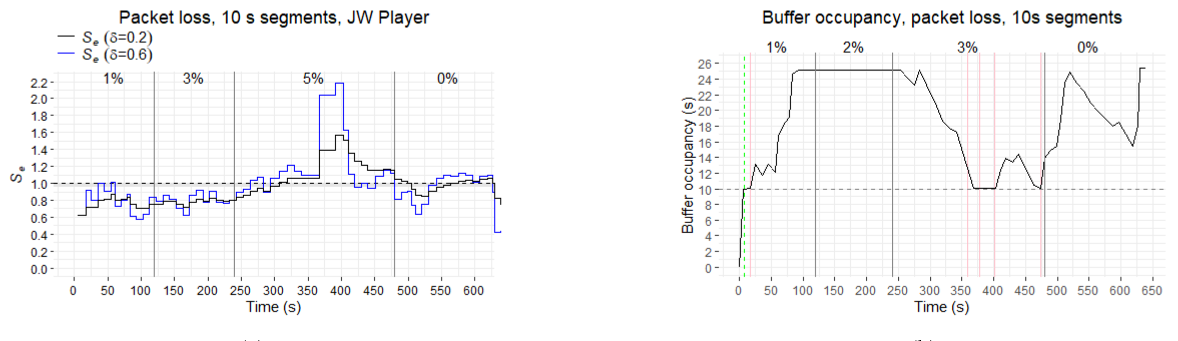
FIGURE 10. Smoothed estimators S_e ($\delta = 0.2$ and $\delta = 0.6$) a) and buffer occupancy b) for JW Player in changing relative bandwidth conditions. The segment length is 10s and the maximum playback buffer size is 25s.

examines both the behavior of estimators with longer segments and the effects of segment length on QoE influence factors.

The changing bandwidth and packet loss condition, realized in sections V-A and V-B, were repeated with a 10-second segment length. Since there were no significant differences in buffer handling between players, and switching algorithms were not examined, only JW Player was used in these tests. Fig. 9a shows how the lengthening of segment duration increases the smoothing of estimates T_e and A_e; that is, it slows down reacting on throughput changes, and the variation decreases. The time for T_e to reach the real throughput may be too long for most real-life use cases. This should be considered when choosing the smoothing factor δ. In contrast, when the packet loss is induced in Fig. 9b, T_e smooths down variation to the level that would prevent bitrate oscillation.

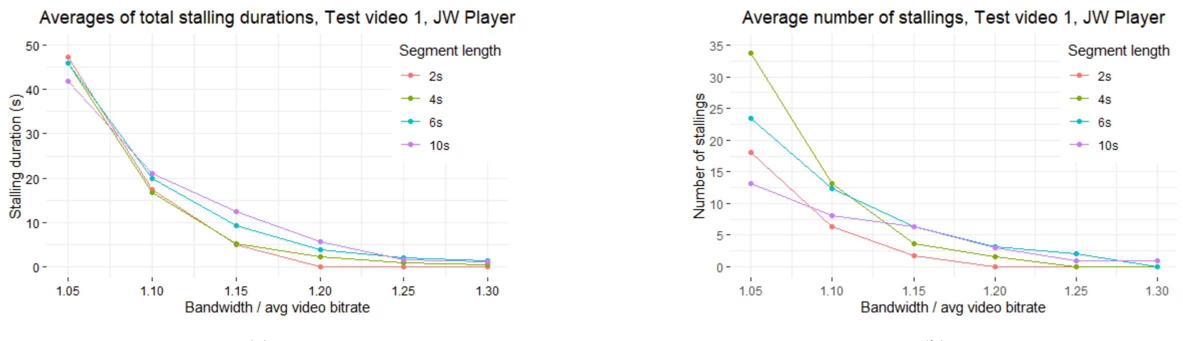
Fig. 10 shows the estimator S_e and buffer occupancy for playback in the 10s length segment duration for varying bandwidth levels and packet loss rates. In this setup, the estimator S_e is formed using two different smoothing factor values, $\delta = 0.2$ and $\delta = 0.6$. JW Player starts the playback right after the first segment has arrived and uses the same maximum buffer size (25s) as with the 2-second segment. Estimator S_e reveals that data are received more slowly than they are removed from the playback buffer in the third and fourth bandwidth periods. The accurate buffer occupancy monitoring reports six stalling occurrences. Segment lengthening from 2 to 10 seconds could not prevent buffer underrun. Corresponding, figures of S_e and buffer occupancy for packet loss test are shown in fig. 11.

Finally, the effect of bandwidth limitation on two QoE influence factors with four different segment lengths (2, 4, 6, and 10s) is depicted in Fig. 12. In these four video variants, keyframes only appear at the start of each segment. Thus, lengthening segments cause decrease in average bitrate. On each relative bandwidth, the video was played five times and stalling occurrences and the total time spent on them were monitored. In Fig. 12a, the total stalling durations during the 10-minute video for JW Player are shown between bandwidths of $1.05 \cdot average\ (video\ bitrate)$ to $1.3 \cdot average\ (video\ bitrate)$. In the bandwidth of $1.05 \cdot average\ (video\ bitrate)$, video stalls on average lasted 47 seconds with 2-second segments and 42 seconds with 10-second segments in total. In a 10-minute-long video, the difference is hardly significant to the user. However, it is interesting that the longest and shortest segment lengths tested had the least stalling events. Closer inspection revealed that, unlike with longer segments, JW Player collected 2-second segments more than one before resuming playback after stalling. It is also possible that some stalling occurrences with 2-second segments were shorter than 0.5 seconds, which was our criterion for stalling. Most stalling occurrences on average (34) were observed with the 4-second segment length in the lowest available bandwidth conditions tested.



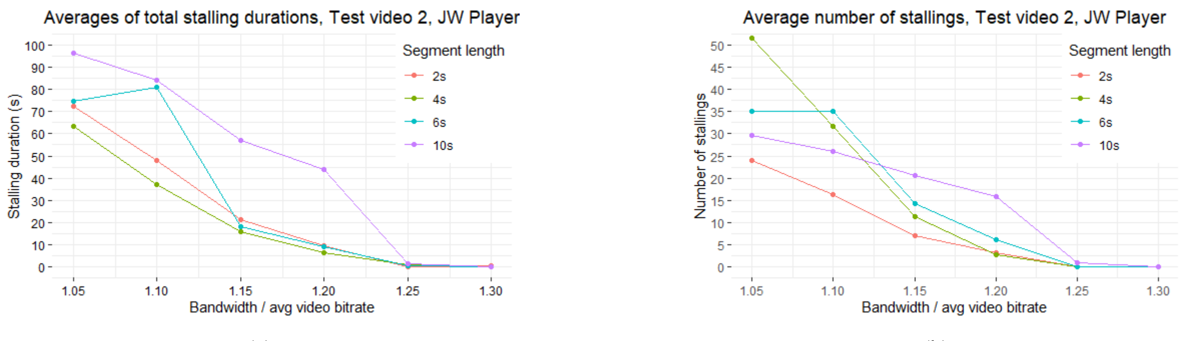
(a) (b)

FIGURE 11. Smoothed estimators S_e ($\delta = 0.2$ and $\delta = 0.6$) a) and buffer occupancy b) for JW Player in changing packet loss conditions. The segment length is 10s and the maximum playback buffer size is 25s. The available bandwidth is 1.3 times the average video bitrate.



(a) (b)

FIGURE 12. Averages of total stalling durations a) and the number of stalling instances b) at different bandwidth levels with four segment lengths. Each test setup is repeated five times. The maximum playback buffer size, in media time, is 25s.



(a) (b)

FIGURE 13. Total stalling durations a) and the number of stalling instances b) at different bandwidth levels with four segment lengths. Each test setup is repeated at least five times. The maximum playback buffer size, in media time, is 25s.

The results were also checked with Test video 2. Test video 2 is smaller, average bitrate varying from 1.83 to 1.91 Mbit/s between segment length versions. Streaming this video clip, gave us more unsteady results that are depicted in Fig. 13. When decreasing bandwidth with longer segment variations (6s and 10s), the playback showed stallings, not caused by buffer underrun. These can result from for example browser settings, device glitch or corrupt video software. The phenomena is demonstrated in Fig. 14. The dark red lines depict the stalling times caused by buffer underrun. Lighter ones are stallings times caused by other reasons. Adding more key frames did not clear up this buffering behavior. However, Video.js player gave consistent results with the Test video 2. These are shown in Fig. 15.

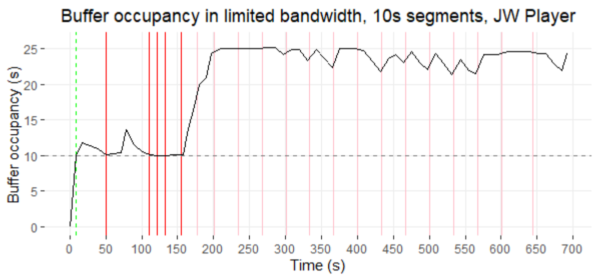


FIGURE 14. A demostration of stalling events when there is no buffer underrun. (Light red lines).

VI. DISCUSSION

Above, the features of four estimators were brought out to evaluate their applicability in HAS algorithms. The smoothed

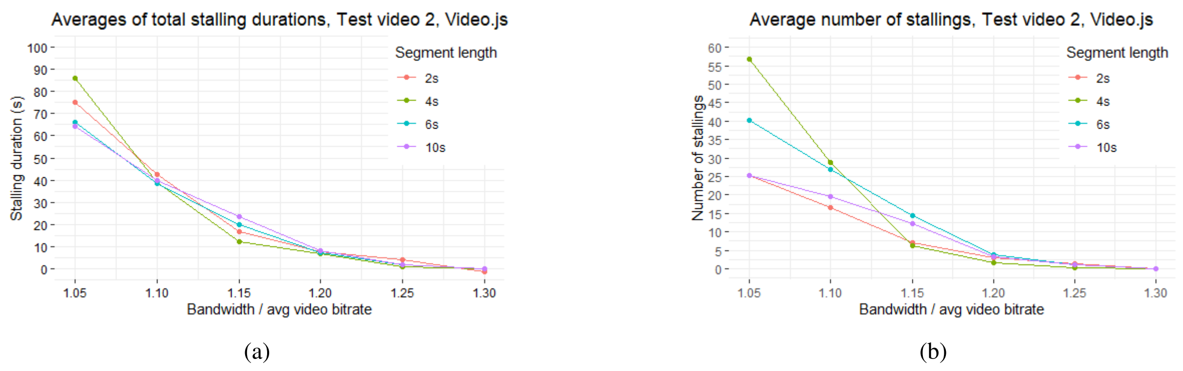


FIGURE 15. Total stalling durations a) and the number of stalling instances b) at different bandwidth levels with four segment lengths using Video.js player. Each test setup is repeated three times. The maximum playback buffer size, in media time, is 25s.

estimator T_e and buffer occupancy are possibly the most used (alone or together). The estimator T_e evaluates absolute throughput to the next segment request interval. To select the most suitable bitrate, the value of T_e is compared with encoding rates that are informed in the playlist file, or for example, to the measured average bitrate of a given number of previous segments. In cases where T_e varies a lot, it is not optimal for the adaptation algorithm. This may happen, for instance, in a congested channel, when T_e has a constant smoothing factor. On the other hand, the video sections that differ a lot from informed target bitrate, may lead to incorrect decisions of segment quality selection if only T_e is used.

Often, the throughput estimator is completed with other metrics. To make T_e more applicable for VBR videos, it was observed in relation to the download estimator A_e, and the difference $\Delta_e = T_e - A_e$. Estimator Δ_e discerns how much of the maximum available bandwidth capacity is unused. When the player fills its playback buffer, maximum utilization is applied. This happens right at the beginning of a streaming session during the initial buffering. In the steady state, the player has idle periods as the playback buffer only has room for a new segment when data are removed from it to play back. Together, T_e and Δ_e are able to tell whether the evaluated throughput is sufficient for the current bitrate, and in theory, estimate the amount by which the video bitrate can be raised or reduced.

Alongside throughput estimators, buffer occupancy is often used in adaptive algorithms by setting different threshold values for it. These values define when to start playback and switch quality level. When used alone, the appropriate segment bitrate selection is not based on any throughput measurements but has to be done for example one quality level at a time. This can cause for example too slow reacting when the buffer is full.

To observe the relative throughput, we simply compared the segment download time to its assumed playback time (segment duration). Estimator S_e is a smoothed version of that metric. The approach combines features from buffer occupancy and throughput estimation. The variation of S_e results both from throughput and segment size changes. A simple goal of the adaptation algorithm using S_e may be selecting a segment that keeps S_e right below the set threshold value λ. A more ambitious algorithm would allow exceeding λ occasionally. The time possible to spend above the threshold value depends on the buffer occupancy development. Buffer occupancy can be obtained by recording in- and outgoing media seconds during S_e tracking. Estimator S_e also tells whether the video bitrate should be raised or reduced.

The behavior of estimators was tested with two players—a commercial JW Player and open source Video.js. Since the HTML5-versions of these players did not express any resource-dependent adapting buffering methods, players' performance were very similar. Only difference was observed when long segments (6s and 10s) where played back in insufficient bandwidth conditions. This caused JWPlayer to pause playback occasionally before buffer underrun. In addition, as a default, the players collected only one segment before starting playback. This resulted in similar initial buffering behavior. As the estimators in this paper predict the metrics for the next request interval, lengthening the segment duration smooths out the estimates; this requires adjusting parameters. Otherwise, changing the segment length hardly affected the QoE metrics observed with JW Player.

VII. CONCLUSION

This paper presented metrics that can be used as building blocks for adaptive algorithms of HAS. We explored four different approaches to evaluating streaming conditions from a client's side. A commonly used test video was played in different bandwidths and throughput conditions, both altering them in the middle of the playback and using the same channel conditions throughout the video. This study also brought out challenges that VRB encoding causes to streaming condition estimators. When the encoding bitrate varies, it is often also with the metrics evaluating streaming conditions. Thus, adaptation algorithms should be designed to handle and interpret fluctuating metrics.

The ability of estimators to predict influence factors of QoE were evaluated. Stalling frequency and total stalling time were used as response variables. Estimator S_e may be the most applicable in predicting streaming conditions independently. Throughput estimator T_e usually needs additive information. The work in this paper also demonstrated that lengthening segments is not a straightforward solution for increasing QoE, even when the initial delay is not considered. The adaptation algorithms of players tested in this paper do not include specific buffering methods. Thus, the true differences of players will probably manifest only in a multi-bitrate environment. Future work will consider testing the S_e estimator as part of an adaptive algorithm. In addition, more precise information on video complexity should be utilized to better optimize the bitrate switching.

REFERENCES

[1] Cisco, "Cisco visual networking index (VNI), Complete forecast update, 2017–2022," San Jose, CA, USA, Tech. Rep. c11-741490-00, 2018.

[2] Y. Chen, K. Wu, and Q. Zhang, "From QoS to QoE: A tutorial on video quality assessment," *IEEE Commun. Surveys Tuts.*, vol. 17, no. 2, pp. 1126–1165, 2nd Quart. 2015.

[3] ITU-T Study Group 12, "Definition of quality of experience (QoE), liaison statement," ITU, Geneva, Switzerland, Tech. Rep. ITU-T FG IPTV, 2007.

[4] Apple Inc. *HLS Authoring Specification for Apple Devices*. Accessed: Nov. 30, 2021. [Online]. Available: http_live_streaming/hls_authoring_specification_for_apple_devices

[5] W. Hu, Z. Wang, and L. Sun, "A measurement study of TCP performance for chunk delivery in DASH," 2016, *arXiv:1607.01172*.

[6] T.-Y. Huang, N. Handigol, B. Heller, N. McKeown, and R. Johari, "Confused, timid, and unstable: Picking a video streaming rate is hard," in *Proc. ACM Conf. Internet Meas. Conf.-(IMC)*. New York, NY, USA: ACM, 2012, pp. 225–238, doi: 10.1145/2398776.2398800.

[7] J. Xue, D.-Q. Zhang, H. Yu, and C. W. Chen, "Assessing quality of experience for adaptive HTTP video streaming," in *Proc. IEEE Int. Conf. Multimedia Expo Workshops (ICMEW)*, Jul. 2014, pp. 1–6.

[8] L. García, J. Lloret, C. Turro, and M. Taha, "QoE assesment of MPEG-DASH in polimedia e-learning system," in *Proc. Int. Conf. Adv. Comput., Commun. Informat. (ICACCI)*, Sep. 2016, pp. 1117–1123.

[9] P. K. Yadav, A. Shafiei, and W. T. Ooi, "QUETRA: A queuing theory approach to DASH rate adaptation," in *Proc. 25th ACM Int. Conf. Multimedia*, Oct. 2017, pp. 1130–1138.

[10] N. K. Karn, H. Zhang, F. Jiang, R. Yadav, and A. A. Laghari, "Measuring bandwidth and buffer occupancy to improve the QoE of HTTP adaptive streaming," *Signal, Image Video Process.*, vol. 13, no. 7, pp. 1367–1375, Oct. 2019.

[11] G. Tian and Y. Liu, "Towards agile and smooth video adaptation in dynamic HTTP streaming," in *Proc. 8th Int. Conf. Emerg. Netw. Exp. Technol.-(CoNEXT)*. New York, NY, USA: ACM, 2012, pp. 109–120, doi: 10.1145/2413176.2413190.

[12] H. T. Le, H. N. Nguyen, N. P. Ngoc, A. T. Pham, and T. C. Thang, "A novel adaptation method for HTTP streaming of VBR videos over mobile networks," *Mobile Inf. Syst.*, vol. 2016, pp. 1–11, Jun. 2016, doi: 10.1155/2016/2920850.

[13] R. Dubin, O. Hadar, and A. Dvir, "The effect of client buffer and MBR consideration on DASH adaptation logic," in *Proc. IEEE Wireless Commun. Netw. Conf. (WCNC)*, Apr. 2013, pp. 2178–2183.

[14] A. Sideris, E. Markakis, N. Zotos, E. Pallis, and C. Skianis, "MPEG-DASH users' QoE: The segment duration effect," in *Proc. 7th Int. Workshop Quality Multimedia Exper. (QoMEX)*, May 2015, pp. 1–6.

[15] M. I. Islam and J. I. Khan, "Video splicing techniques for P2P video streaming," in *Proc. IEEE 35th Int. Conf. Distrib. Comput. Syst. Workshops*, Jun. 2015, pp. 72–76.

[16] C. Liu, I. Bouazizi, and M. Gabbouj, "Segment duration for rate adaptation of adaptive HTTP streaming," in *Proc. IEEE Int. Conf. Multimedia Expo*, Jul. 2011, pp. 1–4.

[17] D. M. Nguyen, L. B. Tran, H. T. Le, N. P. Ngoc, and T. C. Thang, "An evaluation of segment duration effects in HTTP adaptive streaming over mobile networks," in *Proc. 2nd Nat. Found. Sci. Technol. Develop. Conf. Inf. Comput. Sci. (NICS)*, Sep. 2015, pp. 248–253.

[18] A. Mondal, S. Sengupta, B. R. Reddy, M. J. Koundinya, C. Govindarajan, P. De, N. Ganguly, and S. Chakraborty, "Candid with YouTube: Adaptive streaming behavior and implications on data consumption," in *Proc. 27th Workshop Netw. Operating Syst. Support Digit. Audio Video (NOSSDAV)*. New York, NY, USA: ACM, 2017, pp. 19–24, doi: 10.1145/3083165.3083177.

[19] C. Liu, I. Bouazizi, and M. Gabbouj, "Rate adaptation for adaptive HTTP streaming," in *Proc. 2nd Annu. ACM Conf. Multimedia Syst.* New York, NY, USA: ACM, Feb. 2011, pp. 169–174, doi: 10.1145/1943552.1943575.

[20] T. C. Thang, H. T. Le, A. T. Pham, and Y. M. Ro, "An evaluation of bitrate adaptation methods for HTTP live streaming," *IEEE J. Sel. Areas Commun.*, vol. 32, no. 4, pp. 693–705, Apr. 2014.

[21] S. Akhshabi, A. C. Begen, and C. Dovrolis, "An experimental evaluation of rate-adaptation algorithms in adaptive streaming over HTTP," in *Proc. 2nd Annu. ACM Conf. Multimedia Syst.* New York, NY, USA: ACM, Feb. 2011, pp. 157–168, doi: 10.1145/1943552.1943574.

[22] Longtail Ad Solutions. *JW Player*. Accessed: Mar. 1, 2019. [Online]. Available: https://www.jwplayer.com/

[23] Brightcove. *Video.js*. Accessed: Apr. 1, 2019. [Online]. Available: https://videojs.com/

[24] M. Sridharan, K. Tan, D. Bansal, and D. Thaler, *Compound TCP: A New TCP Congestion Control for High-Speed and Long Distance Networks*. Network Working Group, Internet Draft, 2008. Accessed: Nov. 1, 2021. [Online]. Available: https://datatracker.ietf.org/doc/html/draft-sridharan-tcpm-ctcp-02

[25] Apposite Technologies. *Linktropy WAN Emulator*. Accessed: Apr. 1, 2019. [Online]. Available: https://apposite-tech.com/products/linktropy/

[26] Wireshark Foundation. *Wireshark*. Accessed: Nov. 1, 2021. [Online]. Available: https://www.wireshark.org/

[27] Blender Foundation. *Big Buck Bunny*. Accessed: May 14, 2021. [Online]. Available: https://peach.blender.org/

[28] Blender Foundation. *Elephants Dream*. Accessed: May 14, 2021. [Online]. Available: https://orange.blender.org/

[29] Adobe Systems Inc., "HTTP dynamic streaming specification, version 3.0 FINAL," Adobe Syst. Incorporated, San Jose, CA, USA, 2013.

[30] C. Grimm and G. Schlüchtermann, *IP Traffic Theory and Performance*. Berlin, Germany: Springer-Verlag, 2008.

[31] S. Akhshabi, L. Anantakrishnan, A. C. Begen, and C. Dovrolis, "What happens when HTTP adaptive streaming players compete for bandwidth?" in *Proc. 22nd Int. Workshop Netw. Operating Syst. Support Digit. Audio Video-(NOSSDAV)*. New York, NY, USA: ACM, 2012, pp. 9–14, doi: 10.1145/2229087.2229092.

[32] A. Biernacki and K. Tutschku, "Performance of HTTP video streaming under different network conditions," *Multimedia Tools Appl.*, vol. 72, no. 2, pp. 1143–1166, Sep. 2014.

[33] S. Lederer. (2015). *Optimal Adaptive Streaming Formats MPEG-DASH & HLS Segment*. [Online]. Available: https://bitmovin.com/mpeg-dash-hls-segment-length/

SANNA LAINE received the M.Sc. degree in mathematics from the University of Oulu, Finland, in 2006, and the M.Sc. degree in computer science from the University of Jyväskylä, Finland, in 2013. Since 2013, she has been a Project Researcher with Kokkola University Consortium Chydenius (KUC), University of Jyväskylä. Her current research interests include streaming media and education technologies.

ISMO HAKALA (Member, IEEE) received the M.Sc., Ph.Lic., and Ph.D. degrees in mathematics from the University of Oulu, Finland. He joined with the University of Jyväskylä, Finland, in 1999. He is currently a Professor in computer science with Kokkola University Consortium Chydenius (KUC), University of Jyväskylä. He is also the Vice-Director of KUC and the Head of the Information Technology Unit, KUC. His research has focused on intelligent and autonomous sensor systems and he leads the research group in this field. His current research interests include wireless sensor networks with emphasis on data processing, programming frameworks and middleware, localization, and the design and performance evaluation of wireless sensor systems.