Multimedia Tools and Applications (2024) 83:12561–12595
https://doi.org/10.1007/s11042-023-15516-5

Performance analysis of H2BR: HTTP/2-based segment upgrading to improve the QoE in HAS

Minh Nguyen[1] · Hadi Amirpour[1] · Farzad Tashtarian[1] · Christian Timmerer[1] · Hermann Hellwagner[1]

Received: 8 July 2021 / Revised: 29 July 2022 / Accepted: 19 April 2023 /
Published online: 11 July 2023

Abstract

HTTP Adaptive Streaming (HAS) plays a key role in over-the-top video streaming with the ability to reduce the video stall duration by adapting the quality of transmitted video segments to the network conditions. However, HAS still suffers from two problems. First, it incurs variations in video quality because of throughput fluctuation. Adaptive bitrate (ABR) algorithms at the HAS client usually select a low-quality segment when the throughput drops to avoid stall events, which impairs the Quality of Experience (QoE) of the end-users. Second, many ABR algorithms choose the lowest-quality segments at the beginning of a video streaming session to ramp up the playout buffer early on. Although this strategy decreases the startup time, clients can be annoyed as they have to watch a low-quality video initially. To address these issues, we introduced the H2BR technique (**HTTP/2-B**ased **R**etransmission) (Nguyen et al. 2020) that utilizes certain features of HTTP/2 (including *server push*, *multiplexing*, *stream priority*, and *stream termination*) for late transmissions of higher-quality versions of video segments already in the client buffer, in order to improve video quality. Although H2BR was shown to enhance the QoE, limited streaming scenarios were considered resulting in a lack of general conclusions on H2BR's performance. Thus, this article provides a profound evaluation to answer three open questions: *(i)* how H2BR's performance is impacted by parameters at the server side (i.e., various encoding specifications), at the network side (i.e., packet loss rate), and at the client side (i.e., buffer size) on the performance of H2BR; *(ii)* how H2BR outperforms other state-of-the-art approaches in different configurations of the parameters above; *(iii)* how to effectively utilize H2BR on top of ABR algorithms in various streaming scenarios. The experimental results show that H2BR's performance increases with the buffer size and decreases with increasing packet loss rates and/or video segment duration. The number of quality levels can negatively or positively impact on H2BR's performance, depending on the ABR algorithm deployed. In general, H2BR is able to enhance the video quality by up to 17% and 14% in scalable video streaming and in non-scalable video streaming, respectively. Compared with an existing retransmission technique (i.e., SQUAD Wang et al., ACM Trans Multimed Comput

✉ Minh Nguyen
minh.nguyen@aau.at

1 Christian Doppler Laboratory ATHENA, Alpen-Adria-Universität Klagenfurt, Universitätsstraße 65-67, 9020, Klagenfurt, Austria

Commun Applic (TOMM) 13(3s): 45, 2017), H2BR shows better results with more than 10% in QoE and 9% in the average video quality.

Keywords HTTP adaptive streaming · DASH · Retransmission · QoE · HTTP/2 · H2BR

1 Introduction

HTTP Adaptive Streaming (HAS) has become the de-facto technology for delivering video over the Internet with high Quality of Experience (QoE) [6, 47]. In HAS, multiple versions are generated from an original video at the server in order to adapt to the network. Each version is then divided into temporal segments of the same duration. To retrieve the video content, the client sends HTTP requests which fetch the most suitable versions for the next segments based on an adaptive bitrate (ABR) algorithm. The ABR algorithm can rely on client parameters such as buffer status and/or network parameters such as throughput.

There are two open research issues of existing ABR algorithms. Firstly, they sometimes lead to video quality variations while adapting the video quality to the network condition (i.e., throughput). When the throughput is unfavorable, ABR methods have to decrease the segments' quality, resulting in quality switches between segments. These quality switches negatively impact the client's QoE [15]. Most ABR algorithms aim to request higher video quality versions when the throughput increases. However, they often ignore improving the quality of already buffered low-quality but not yet played-out segments even though the throughput may be enough to sustain more than one segment delivery at the same time. Meanwhile, a trace containing video streaming information collected from Akamai's video content delivery network (CDN) shows that almost 36% of the sessions suffer at least one downward variation in video quality [8]. Secondly, many state-of-the-art ABR algorithms start a streaming session (i.e., perform the startup phase) with the lowest-quality segments to quickly ramp up the buffer and, as a result, decrease the startup delay. However, clients may quickly terminate a streaming session if the initial video quality is not good enough [12].

We propose the **HTTP/2-B**ased **R**etransmission technique (H2BR) [33], which replaces already buffered low-quality video segments with higher-quality versions while avoiding adversely affecting the performance of the underlying ABR algorithms. H2BR scans the buffer to detect the segments with lower qualities than adjacent ones. Higher-quality versions are determined based on the buffer occupancy and the estimated throughput. To make the retransmitted segments arrive at the client on time and reduce the request overhead, we exploit certain HTTP/2 features, including: *(i) multiplexing*, *(ii) stream priority*, *(iii) server push*, *(iv) stream termination*. The client can send multiple requests to download segments concurrently with the *multiplexing* feature. *Stream priority* helps the client control the bandwidth allocated for simultaneous segments by assigning proper `PRIORITY` weights to these segments. To reduce the number of requests, the *server push* feature is utilized, in which the client retrieves several segments sequentially with a single HTTP GET request. Finally, *stream termination* terminates retransmitted segments with high probabilities of arriving after playout time or when the buffer level is at risk.

The experimental results from our prior work [32, 33] show that H2BR is able to improve the client's QoE. In this article, we address two open issues: *(i)* H2BR was evaluated with different packet loss rates and buffer sizes for a scalable video coding format only in [32]. In [33], only segment duration for non-scalable video coding format was considered. However, a client can encounter different input parameters' values for either video coding format in real-world environments. For example, a video can be streamed to the client with a large or small buffer over a network in which the packet loss rate is low or high. Therefore, we take into account the impact of all of these parameters on H2BR for both non-scalable and scalable video coding formats. In addition, various sets of video representations (termed "*bitrate ladders*" in the following) will be evaluated as the video can be encoded into different bitrate ladders based on the service provider and video content [2]. *(ii)* H2BR was compared to the related work in only some specific configurations in [32, 33].

The objective of this work is to enhance the QoE of ABR algorithms by replacing low-quality segments in the buffer.The contribution of this paper is three-fold:

- **H2BR retransmission technique:** We leverage the HTTP/2 features to enhance the QoE by upgrading low-quality segments stored in the buffer. These HTTP/2 features include *server push*, *multiplexing*, *stream priority*, and *stream termination*.
- **Comprehensive evaluation:** To fully assess H2BR's performance, we conduct various experiments with different experimental configurations that reflect real-world streaming scenarios. In addition, detailed comparisons between H2BR and existing related work are drawn.
- **Guidelines on using H2BR:** We summarize our findings as guidelines for the H2BR usage in various streaming scenarios, including different video formats, bitrate ladders, ABR algorithms, and packet loss rates.

The remainder of the article is structured as follows. Section 2 provides background and related work. The details of the H2BR approach are given in Section 3, followed by the experimental setup in Section 4. Evaluations and guidelines on using H2BR in various scenarios are given in Sections 5 and 6 . Section 7 concludes the article and highlights future work.

2 Background and related work

2.1 HTTP/2-based video streaming

HTTP/2 was released in 2015 by the Internet Engineering Task Force (IETF) [5]. It was issued as a higher performance alternative over its predecessor HTTP/1.1 and evaluated in the context of HAS by Mueller et al. [27]. HTTP/2 provides some key features for HAS such as *server push*, *stream priority*, *multiplexing*, and *stream termination*. *Server push* allows the client to retrieve multiple segments with the same quality by sending a single request. Using the *stream priority* feature, the client can request the server to spend more resources on pushing more important segments when some segments are delivered in parallel by the *multiplexing* feature. The client indicates the higher-priority segments by setting a higher `PRIORITY` weight (i.e., integer values between 1 and 256) for the stream delivering them. The throughput allocated to concurrent segments is proportional to the `PRIORITY` weight. Finally, the *stream termination* feature allows the client to terminate a stream immediately by sending the server an `RST_STREAM` frame associated with that stream [5].

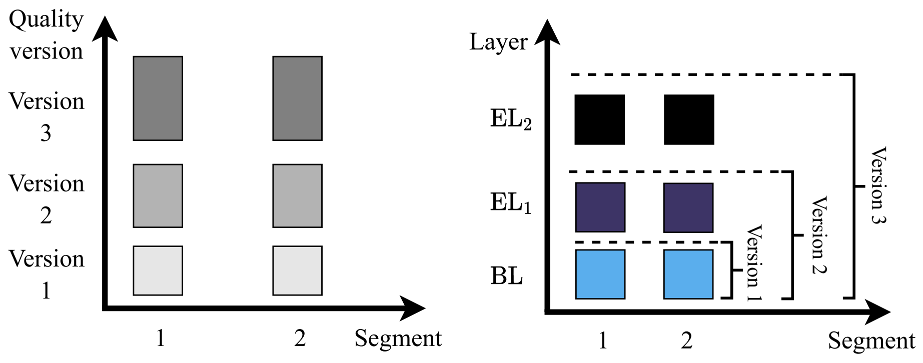
The *server push* feature was first exploited by Wei et al. [50]. The authors implemented a *k-Push* strategy to reduce latency in live streaming by sending requests indicating the required video quality and a fixed value of *k* – the number of consecutive segments to be delivered in response to a single request. However, the fixed number of pushed segments may result in stalls when the buffer is low and the throughput drops, as the client is supposed to receive all of the requested segments before switching to a lower quality version. To address this issue, Nguyen et al. [28] proposed an adaptive method of varying the value of k based on a cost function as a weighted sum of *buffer cost* and *request cost*. The *request cost* is a linearly decreasing function of k, whereas the *buffer cost* is proportional to k.

Stream priority and *stream termination* have shown to be useful in video streaming as well. The methods proposed in [29, 31] utilize *stream priority* to control the arrival time of different tiles in 360° videos. Tiles belonging to the same segment are downloaded simultaneously, and visible tiles with higher qualities are assigned higher priority weights so that the server uses more bandwidth to deliver these tiles. In case some tiles cannot be downloaded before their playout deadlines, *stream termination* will be used to terminate them. These heuristics can cope with the high bandwidth requirement of 360° videos. Similarly, the work in [30] makes use of *stream priority* and *stream termination* to address the viewport estimation errors. Recently, Yahia et al. [53] have presented a client-based algorithm for low-latency video streaming that uses *stream priority* to manage the video frame delivery, and *stream termination* to discard some frames when throughput becomes unfavorable. *Stream termination* is not only utilized for 360° video streaming but also works efficiently in the live streaming context for conventional videos by canceling a segment request if its download may lead to stalls in the future [21].

Compared to the above methods, H2BR focuses on enhancing the video quality of segments already stored at the buffer of the client and operates complementary to existing ABR algorithms rather than making a decision on the bitrate of the next segments to be requested.

2.2 Non-scalable and Scalable Video Coding (SVC) in streaming

In non-scalable video streaming, different versions of a video segment are encoded independently. Each quality version is represented by a single file as shown in Fig. 1(a), and the client chooses a quality version to watch by downloading the corresponding file (or portions thereof) from the server. This strategy leads to increased storage costs and bandwidth usage in the CDN due to the redundancy that exists among those versions [35]. Moreover, non-scalable video streaming does not allow the client to upgrade the video quality after sending segment requests, which may result in an unnecessary low-quality video because of wrong quality decisions.



(a) Non-salable video coding format (b) Scalable video coding format

Fig. 1 Non-scalable and scalable video coding formats

On the other hand, SVC reduces this redundancy and integrates all quality versions into a single bit stream with the ability to extract a specific version. A quality version with the minimum requirements is encoded as a base layer (BL). One or more enhancement layers (ELs) are encoded using the BL as the reference layer. ELs with lower resolutions or qualities can be used to encode higher ELs in addition to the BL. Figure 1(b) illustrates the quality versions in SVC videos. This video quality structure enables the client to add the ELs based on the network conditions so that the video quality can be gradually improved. This will avoid the negative impact of wrong quality decisions of non-scalable video streaming [37]. Additionally, in terms of service delivery, SVC implies lower storage than non-scalable coding [19]. A robust rate adaptation algorithm for scalable video streaming has been proposed in [13]. To overcome the end-to-end delay caused by requesting multiple layers per segment in scalable video streaming, parallel and pipelined downloading of segment layers has been proposed by Bouten et al. [10]. An interest packet control for Named Data Networking (NDN) has been proposed in [36], which reduces unused data packets in scalable video streaming. The performance of scalable video streaming with respect to spatial and quality scalability options is compared to non-scalable video streaming using various encoder implementations in [14].

In this article, we use the video compression standard HEVC, one of the most popular video codecs according to Bitmovin's Video Developer Report 2020 [9], for non-scalable videos. For the SVC videos, SHVC [43, 54], the latest scalable video coding format, is used in this article.

2.3 ABR algorithms and retransmission techniques

ABR schemes have been studied extensively in the literature [6]. These approaches can be classified into *(i) bandwidth-based* (e.g., [25, 28]), *(ii) buffer-based* (e.g., [16, 46]), and *(iii) hybrid* (e.g., [18, 34]).

In the *bandwidth-based* schemes, the client merely relies on the measured network bandwidth (or throughput) to select suitable quality versions. The works in [24, 25, 39] measure the throughput as the downloaded segment's size divided by the download time of that segment. Zhang et al. [56] propose an ABR algorithm, namely piStream, for clients in LTE networks. PiStream estimates the available throughput with a resource monitor module working as a physical-layer daemon. Miller et al. [26] introduce the LOLYPOP ABR algorithm which predicts throughput on different time scales such as 1 second and 10 seconds for the purpose of improving QoE and obtaining low latency. Probe AND Adapt (PANDA) [23] estimates the available bandwidth accurately and tries to address fairness issues by reducing bitrate oscillations in a shared link.

Buffer-based ABR approaches select the video representation based on the buffer occupancy at the client. BBA [16] maps the instant buffer level to an available quality level by an increasing linear function. However, BBA requires a large buffer to avoid throughput fluctuation. Spiteri et al. [46] introduce the BOLA ABR algorithm that takes into account the buffer level, playback utility and playback smoothness to determine the suitable quality level for the next segments. Sieber et al. [45] propose an SVC-based adaptation algorithm called bandwidth-independent efficient buffering (BIEB). BIEB aims to maximize video quality while decreasing the number of quality oscillations and avoiding stalls. BIEB attempts to maintain a stable buffer occupancy before increasing the quality (via enhancement layers). However, BIEB does not consider bitrate switches or stalls in the QoE model during peak times when dynamic cross traffic occurs in the network. Yadav et al. [52] introduce an M/D/1/K queue model for a DASH client to calculate the expected buffer occupancy given a bitrate choice, network throughput, and buffer capacity. Considering a diverse set of scenarios, the authors evaluate QUETRA and show that QUETRA leads to better QoE than the existing algorithms.

In the *hybrid* adaptation methods, the client makes the quality level decision based on both available throughput and current buffer occupancy. SARA [18] splits the buffer size into three regions and selects the suitable video quality based on the available throughput and the region to which the current buffer level belongs. Recently, the work in [34] considers three cost factors of downloading a segment of a specific video quality, including buffer cost, quality cost and throughput cost. Downloading a high-bitrate segment results in high buffer cost and throughput cost because of long delivery time but low quality cost due to high quality. The video quality with the lowest total cost, which is a weighted sum of the three costs, will be selected. Zhou et al. [57] investigate the problem of providing smooth video bitrate switching over multiple servers due to their various bandwidths. They propose a block-based ABR method considering both the bandwidth of multiple servers and the buffer occupancy information. The main objective in this work is to avoid buffer overflow and underflow; however, the authors do not consider other metrics such as switching frequency and amplitude factors. In contrast, in H2BR we provide a retransmission technique that can work on top of ABR algorithms to upgrade the buffered segments and improve the QoE. An Adaptation and Buffer Management Algorithm (ABMA) is proposed in [51]. Using the measured segment download time, ABMA improves the bit-stream switching. ABMA copes with short-term bandwidth variations through tuning the client buffer size. Batalla et al. [3] improve the performance of ABMA in terms of computational cost. However, the authors do not investigate the impact of their approach on the user-perceived quality. Belda et al. [4] introduce an ABR scheme named Look Ahead that considers the bitrate variability of quality versions. Although Look Ahead avoids stalling events, it suffers from low average video quality due to a conservative process of selecting the quality versions for the next segments.

Though the above ABR algorithms show promising results to some extent, they still suffer from the time-varying fluctuation of the throughput. Once they download a low-quality segment, the viewer has to watch it even though the upcoming throughput may be favorable. This negatively impairs the QoE of the viewer. Our idea to solve this research issue is to replace low-quality segments by high-quality ones (i.e., to redownload segments) when the throughput can afford additional segments besides the next segments. However, this strategy raises another question: *"How to deliver those segments effectively to assure the QoE improvement?"* When low-quality segments are about to be played soon, the redownloaded segments need more throughput to arrive at the client before their playback time. Otherwise, the next segments should be devoted more throughput so that the buffer is filled faster to avoid stall events.

We address the aforementioned question by proposing H2BR, which leverages HTTP/2 features, especially *multiplexing* and *stream priority* to control the throughput allocated to multiple segments. H2BR detects low-quality segments in the buffer and replaces them with higher-quality versions (named retransmitted segments) while downloading the next segments concurrently. This enables the end user to watch a higher-quality video.

To show that H2BR can work efficiently on top of various ABR algorithms, in this paper we use four ABR schemes in our experiments: *(i)* for non-scalable video streaming, *AGG* [28], *BBA* [16], and *SARA* [18]; *(ii)* for scalable video streaming, *Backfilling* [37]. These approaches are selected as they belong to various classes of ABR algorithms classified in [6] (i.e., throughput-based, buffer-based, hybrid, and scalable), and are straightforward to implement.

AGG [28] is a throughput-based algorithm and aggressively – hence, the name AGG – selects the maximum bitrate that is lower than the estimated throughput for the next segment to be requested. The experimental results in our previous work [33] showed that the AGG algorithm could provide a relatively high overall QoE score when integrated with the proposed extension component H2BR for delivering non-scalable videos. To exploit this approach for scalable video streaming, the client selects the maximum number of layers for the next segment with the total bitrate lower than the estimated throughput. Then, layers of this segment are fetched one by one starting from the BL to the top EL. This process is repeated for the next segment after all layers of the current segment are downloaded. Figure 2 demonstrates the modified AGG approach for scalable video streaming. There are four segments in the buffer. The first and third segments have two layers (one BL and one EL), and the others have one layer. These layers are downloaded in the order from layer #1 to layer #6, and each of them corresponds to a single request from the client.

BBA [16] is a buffer-based ABR algorithm that uses an increasing function $f(B_i)$ to map the buffer level B_i to a corresponding bitrate. The buffer size is split by two thresholds, r and cu ($r > 0$ and $cu > 0$), where r represents the size of the reservoir and cu denotes the size of the cushion. If the current buffer is less than r, then the next bitrate is the minimum one. When the current buffer is more than $r + cu$, the next bitrate is the maximum one. Otherwise, the value of the function $f(B_i)$ is used to determine the next bitrate.

As a hybrid ABR algorithm, SARA determines the next bitrate as the result of the buffer and the estimated throughput constraints. This algorithm divides the buffer by three thresholds, I, B_α, and B_β ($I < B_\alpha < B_\beta$). If the current buffer is smaller than or equal to I, the minimum bitrate is chosen. When the current buffer is larger than I but not greater than B_α, the bitrate is conservatively increased by one level. In case the buffer is between B_α and B_β, the next bitrate is higher or equal to the current quality version, based on the estimated throughput and current buffer level. Otherwise, the best suitable quality version for the throughput is chosen as the next bitrate, and the client waits for a period to send the next request to avoid unnecessary video downloads if the client suddenly quits the video.

In the Backfilling algorithm [37], the client checks the number of the BL segments in the buffer. If the buffer contains all BLs, the adaptation algorithm upgrades the quality of the segments starting from the segment with the last playout time by downloading its next EL. Otherwise, the BL of the segment with the earliest playout time is downloaded first. Figure 3 presents this approach when the buffer size is five segments. After downloading all the BLs of these segments, the approach starts to download the first EL of the fifth segment,

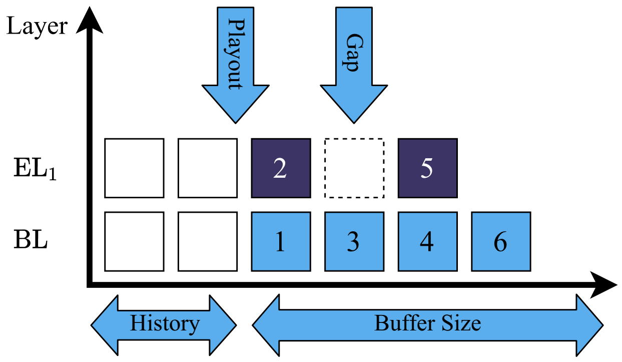


Fig. 2 Modified AGG for scalable video streaming

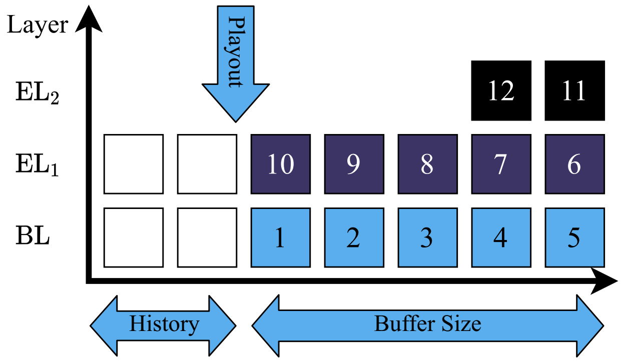


Fig. 3 Backfilling for scalable video streaming

then turns to the fourth segment with its next layer. When the first EL of the first segment is completely downloaded, the second EL of the fifth segment will be fetched, and so on.

3 The proposed HTTP/2-based segment upgrading approach: H2BR

In this section, we introduce a method to measure the network condition when the data from multiple segments are delivered simultaneously, followed by an in-depth description of the H2BR technique. Even though H2BR is originally designed for the non-scalable video coding format, we will show that it is able to work efficiently for scalable videos with some minor enhancements.

3.1 Throughput measurement

The traditional client-side throughput measurement determines the throughput T based on the segment size S of the last segment divided by its download time, i.e., the period from sending an HTTP GET request (at time t^0) to completely receiving the segment (at time t^*) as shown in (1).

$$T = \frac{S}{t^* - t^0}. \tag{1}$$

However, when the server pushes multiple segments simultaneously, (1) merely measures the average throughput allocated for one segment and does not reflect the whole available throughput capacity. To deal with this problem, we propose to divide the amount of the data downloaded between two consecutive end times of segment downloads over this period. Therefore, the throughput T will be computed as

$$T = \frac{D_{t_{n-1}}^{t_n}}{t_n - t_{n-1}}, \tag{2}$$

where t_n is the n-th time that the client completely downloads a segment, and $D_{t_{n-1}}^{t_n}$ is the amount of the data downloaded between t_{n-1} and t_n. $D_{t_{n-1}}^{t_n}$ can be inferred from `DATA` frames, and the `END_STREAM` flag provides t_n. If there is a negligible period between t_{n-1} and t_n, (i.e., when $t_n - t_{n-1} \ll \tau$, where τ denotes the segment duration), computing (2) is skipped at timestamp t_n.

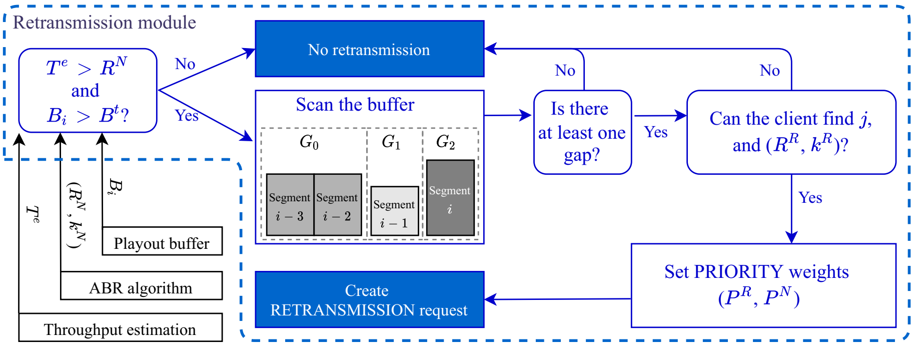


Fig. 4 Overview of the proposed retransmission technique H2BR

3.2 H2BR for non-scalable video streaming

Currently, most of the ABR algorithms try to optimize the QoE by considering the bitrate of the next segments and ignoring the segments located in the buffer [6]. In this section, we introduce the H2BR technique, whose overview is depicted in Fig. 4, to improve the QoE by replacing low-quality segments currently stored in the buffer with higher-quality versions. This technique is executed by an additional component, namely the *Retransmission module*, as illustrated in Fig. 5. The notations used in our H2BR approach are described in Table 1.

This proposed component fetches the information from the *ABR algorithm*, *Playout buffer*, and *Throughput estimation* to determine which segments at what bitrates and priority weights should be retransmitted. In the end, the client sends a *NEXT request* to get the (multiple) next segments and, if needed, a *RETRANSMISSION request* for retransmitted ones.

The process of the proposed technique is described as follows. Given that $\mathcal{V}$ is the set of quality versions, the last downloaded segment is segment i, and the current buffer after the download of segment i is B_i. Let T^e denote the estimated throughput. The *ABR algorithm* component provides the bitrate of the next segments, $R^{\mathcal{N}}$, and the number of the next segments, $k^{\mathcal{N}}$. The *Retransmission module* determines the following parameters: $(R^{\mathcal{R}}$,

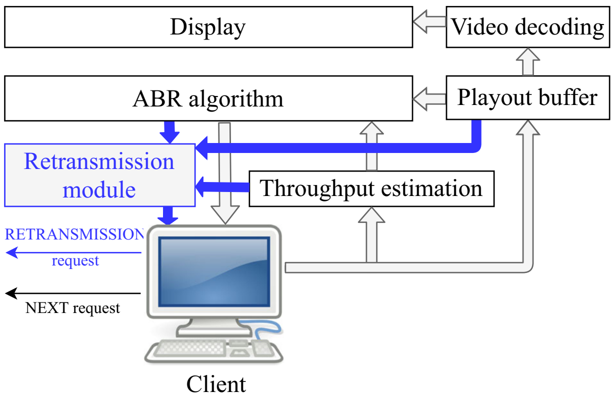


Fig. 5 The *Retransmission module* and other extensions at the client

Table 1 Notations used in our H2BR algorithm

Notation	Description
T	Measured throughput
T^e	Estimated throughput
$T^{\mathcal{R}}$, $T^{\mathcal{N}}$	The throughput allocated to retransmitted and next segments
B_i	Buffer occupancy after downloading segment i
B^e	The estimated buffer after downloading the retransmitted and next segments
B^t	Buffer threshold to allow retransmission
$R^{\mathcal{R}}$, $R^{\mathcal{N}}$	The bitrate of retransmitted and next segments, respectively
R_i^l, R_i^h	The bitrate of the lower-quality and higher-quality of adjacent groups of segment i
$k^{\mathcal{R}}$, $k^{\mathcal{N}}$	The number of segments downloaded in the retransmission and next requests, respectively
$P^{\mathcal{R}}$, $P^{\mathcal{N}}$	The PRIORITY weights of the retransmitted and next segments
G_i	The ith group of adjacent segments with the same quality
ϕ_{G_n}, Φ_{G_n}	The smallest and largest indices of segments in group G_n
$g(i)$	The group index of segment i
t_i^a	The available download time for segment i
t_j^p	The playback time of segment i
t^c	The time of the currently played segment

$k^{\mathcal{R}}$), ($P^{\mathcal{R}}$, $P^{\mathcal{N}}$) and j. $R^{\mathcal{R}}$ is the bitrate of the segments to be retransmitted, $k^{\mathcal{R}}$ is their number, and j is the index of their first segment, which means the retransmitted segments are $\{j, j+1, ..., j+k^{\mathcal{R}}-1\}$. $P^{\mathcal{N}}$ and $P^{\mathcal{R}}$ are the PRIORITY weights of the next and retransmitted segments, respectively. The retransmission is triggered when the estimated throughput T^e is larger than the bitrate of the next segments $R^{\mathcal{N}}$, and the current buffer B_i is higher than a threshold B^t. Afterwards, all the segments which are being stored in the *Playout buffer* will be scanned to search for gaps, which are defined as lower-quality segments compared to their adjacent ones, and the proposed component groups all the same-quality consecutive segments together. Let G_n denote the nth group in the buffer and ϕ_{G_n} and Φ_{G_n} denote the smallest and largest indices of segments in group G_n, respectively. The group that segment i belongs to is indicated by $g(i)$. An example is given in Fig. 4. The buffer consists of four segments, and both segments $(i-3)$ and $(i-2)$ belong to the group 0, thus $g(i-3) = g(i-2) = G_0$. The $(i-1)$th and ith segments with different quality versions are put in groups G_1 and G_2, respectively.

In our proposed method, $R^{\mathcal{R}}$, $k^{\mathcal{R}}$ and j must satisfy three conditions. First, segment j needs to completely arrive in the client before its playout time to ensure it can be decoded. The available download time of segment j can be computed as

$$t_j^a = t_j^p - t^c, \tag{3}$$

where t_j^p denotes the playback time of segment j, and t^c is the time of the currently played segment. The throughput assigned to deliver retransmitted segments, $T^{\mathcal{R}}$, must be high enough to download segment j within period t_j^a, but less than the estimated throughput

T^e so that the next segments can also be delivered. $T^{\mathcal{R}}$ is represented by

$$T^{\mathcal{R}} = \frac{R^{\mathcal{R}} \times \tau}{t_j^a}. \tag{4}$$

Second, the estimated buffer B^e after downloading all the next and retransmitted segments must be higher than a threshold Θ to avoid effects on the next-bitrate decision of the *ABR algorithm* component. Θ is chosen based on the current buffer B_i and buffer thresholds used in the ABR algorithm. As an example, assume that the current buffer level is equal to 15 s and, if there were no retransmission, the future buffer level would go down to 12 s. However, the retransmission could reduce this to 8 s since more time is needed to download retransmitted segments. In case the ABR algorithm chooses a lower-quality version when the buffer level falls below 10 s, for instance, the retransmission negatively affects the decision of the ABR algorithm. Therefore, in this case, Θ is set to 10 s to prevent this effect. The estimated buffer B^e is based on the current buffer. Right after $(k^{\mathcal{N}} + k^{\mathcal{R}})$ segments are completely downloaded, only the $k^{\mathcal{N}}$ next segments contribute $k^{\mathcal{N}} \times \tau$ seconds to the buffer. Meanwhile, the buffer is drained by the download time of $(k^{\mathcal{N}} + k^{\mathcal{R}})$ segments, which is $\frac{k^{\mathcal{N}} \times \tau \times R^{\mathcal{N}} + k^{\mathcal{R}} \times \tau \times R^{\mathcal{R}}}{T^e}$. Therefore, B^e is determined as follows:

$$B^e = B_i + k^{\mathcal{N}} \times \tau - \frac{k^{\mathcal{N}} \times \tau \times R^{\mathcal{N}} + k^{\mathcal{R}} \times \tau \times R^{\mathcal{R}}}{T^e}. \tag{5}$$

Third, as our objective is to fill the gaps, the bitrate of retransmitted segments does not need to be too high; it is sufficient to be in the range of qualities (bitrates) of the adjacent groups of segment j. Let R_j^l and R_j^h be the bitrates of the lower-quality and the higher-quality adjacent groups, respectively. Besides, the number of retransmitted segments is not more than the number of segments in group $g(j)$. Therefore, $(R^{\mathcal{R}}, k^{\mathcal{R}})$ and j are determined to satisfy the following constraints:

$$\begin{cases} T^{\mathcal{R}} < T^e \\ B^e \geq \Theta \\ R_j^l \leq R^{\mathcal{R}} \leq R_j^h \\ k^{\mathcal{R}} \leq \Phi_{g(j)} - \phi_{g(j)} + 1 \end{cases} \tag{$*$}$$

If there are multiple pairs $(R^{\mathcal{R}}, k^{\mathcal{R}})$, we will prioritize the one having smaller $g(j)$ and larger $k^{\mathcal{R}}$. If both the NEXT and RETRANSMISSION requests are sent by using merely the HTTP/2 *multiplexing* feature as in SQUAD [8], the data belonging to each request will utilize half of the whole capacity of the network. This could result in unfairness when the retransmitted segments need more throughput because of their larger data amount or limited available time for delivery. Therefore, finally, the *Retransmission module* determines the `PRIORITY` weights for the next and retransmitted segments to make sure that the retransmitted ones are assigned enough throughput $T^{\mathcal{R}}$. These parameters are based on

$$\frac{P^{\mathcal{R}}}{P^{\mathcal{N}}} = \frac{T^{\mathcal{R}}}{T^e - T^{\mathcal{R}}}. \tag{7}$$

Our general technique is summarized in Algorithm 1. It should be noted that, according to RFC 7540 [5], $P^{\mathcal{R}}$ and $P^{\mathcal{N}}$ are integers between 1 and 256. We propose to determine the integer pair $(P^{\mathcal{R}}, P^{\mathcal{N}})$ according to Algorithm 2. Firstly this algorithm sets the bigger parameter to the maximum value, i.e., 256, then the other parameter is straightforwardly

```
Input  : $T^e$, $B_i$, $(R^{\mathcal{N}}, k^{\mathcal{N}})$
Output: $(R^{\mathcal{R}}, k^{\mathcal{R}})$, $(P^{\mathcal{R}}, P^{\mathcal{N}})$, $j$
if $B_i > B^t \land T^e > R^{\mathcal{N}}$ then
    for each segment in the buffer do
        Group same-quality consecutive segments: $G_n \leftarrow \{\phi_{G_n}, \phi_{G_n} + 1, ..., \Phi_{G_n}\}$;
    end
    for each group $G_n$ do
        Find $(R^{\mathcal{R}}, k^{\mathcal{R}})$ and $j$ following $(*)$;
    end
    if the client can find $(R^{\mathcal{R}}, k^{\mathcal{R}})$ and $j$ then
        Compute $(P^{\mathcal{R}}, P^{\mathcal{N}})$ according to Algorithm 2;
        Create RETRANSMISSION request;
    else
        No retransmission;
    end
else
    No retransmission;
end
```

Algorithm 1 HTTP/2-based retransmission technique.

calculated based on (7). If they are equal, both are set to 1. The `PRIORITY` weights $P^{\mathcal{R}}$ and $P^{\mathcal{N}}$ are added in the `Weight` field of the `HEADERS` frames [5] that open the streams.

During the retransmission period, the *Retransmission module* monitors the instant buffer status and the time remaining before the retransmitted segment is played out. If it detects the current buffer is less than a certain threshold B^l or the remaining retransmission time is less than a pre-defined threshold t^l, all the retransmitted segments that have not fully arrived in the client will be terminated by an `RST_STREAM` frame of HTTP/2 sent from the client. This frame includes an error code to indicates the reason why the stream is terminated. The

```
Input  : $T^{\mathcal{R}}$, $T^e$
Output: $P^{\mathcal{R}}$, $P^{\mathcal{N}}$
$proportion \leftarrow \frac{T^{\mathcal{R}}}{T^e - T^{\mathcal{R}}}$;
if $proportion < 1$ then
    $P^{\mathcal{N}} = 256$;
    $P^{\mathcal{R}} = \lfloor P^{\mathcal{N}} \times proportion \rfloor$;
else if $proportion > 1$ then
    $P^{\mathcal{R}} = 256$;
    $P^{\mathcal{N}} = \left\lfloor \frac{P^{\mathcal{R}}}{proportion} \right\rfloor$;
else
    $P^{\mathcal{R}} = 1$;
    $P^{\mathcal{N}} = 1$;
end
```

Algorithm 2 Determination of priority weights.

list of error codes can be found in [5]. In our case, we use the code `CANCEL` with the value `0x8` to express that the stream delivering retransmitted segments is no longer needed. To avoid frequent termination of retransmission, both B^l and t^l should be small, but B^l must be high enough to minimize the risk of playout stalls and to start a new retransmission session quickly. Therefore, in our experiments we set $t^l = 0.1$ s, $B^l = B/4$, and $B^t = B/2$, where B denotes the buffer size. The values of parameters of the H2BR and ABR algorithms in our experiments are defined in Table 2, and the buffer size is changed as shown in Table 5.

An example of the performance of H2BR in a cascade network trace according to [7] is illustrated in Fig. 6. It can be seen that when the throughput becomes more favorable (from second 120 on), our proposed method can re-download many better-quality segments to substitute for low-quality ones stored in the buffer to provide higher video quality.

3.3 H2BR for scalable video streaming

As shown in Fig. 2, there might be some quality gaps in the buffer while streaming scalable videos. To cope with this issue, some ELs can be downloaded to fill those gaps with some specific conditions. For example, as shown in Fig. 7 in which there are four segments in the buffer, if the throughput that delivers the BL of segment 4 (#6) is high enough to sustain another additional layer (i.e., #7 of segment 2), then the client should download both layers to solve the gap issue. This idea can be applied by our proposed method H2BR.

To deploy H2BR for scalable video streaming, instead of retransmitting segments, H2BR additionally downloads higher layers for the segments that cause quality gaps. As a high layer is rendered to be displayed only when all of its lower layers are available at the client, H2BR needs to download the one-level-higher layer to upgrade the low-quality segment. Thus, the $R^{\mathcal{R}}$ is simply the bitrate of the next enhancement layer. Besides, because each request only includes the information of one layer of a segment, the *server push* feature is not suitable in this case. Consequently, the number of the next layers $k^{\mathcal{N}}$ and the number of retransmitted layers $k^{\mathcal{R}}$ are both set to 1. In other words, H2BR does not use the *server push* feature in scalable video streaming (i.e., $k^{\mathcal{R}} = 1$), and $R^{\mathcal{R}}$ is straightforward to determine.

4 Evaluation setup

In this article, we consider the performance of H2BR in two experimental setups to answer the aforementioned research questions, respectively:

- H2BR under different conditions to evaluate the impact of individual parameters (i.e., bitrate ladders, segment duration, packet loss rates, and buffer sizes) on the performance of our proposed method.
- H2BR under a given configuration compared to related work.

4.1 Bandwidth trace

In the previous paper [33], we evaluated H2BR's performance in a low-throughput network. Therefore, in this paper, we use a higher-throughput network trace collected in [48] on a bus ride, as illustrated in Fig. 8. The average bandwidth in this trace is 19822 kbit/s with a standard deviation of 11240 kbit/s, and the maximum bandwidth is 64143 kbit/s. It should be noted that our test sequence (see later) has the maximum bitrate of 20000 kbit/s and,

Table 2 Parameters used in the experiments

Parameters	ABR algorithms and their internal parameters							
	AGG	BBA ($r = 10$ s, $cu = 30$ s, $B = 44$ s)			SARA ($I = 14$ s, $B_\alpha = 20$ s, $B_\beta = 30$ s, $B = 34$ s)			
		$B_i < r$	$r \leq B_i < r + cu$	$B_i \geq r + cu$	$B_i < I$	$I \leq B_i < B_\alpha$	$B_\alpha \leq B_i < B_\beta$	$B_i \geq B_\beta$
Θ	$B/4$	r	$f^{-1}(R_M)^{(**)}$	$r + cu$	I	$B_\alpha/2$	B_α	B_β
B^t					$B/2$			
B^l					$B/4$			
t^l					0.1 s			

$(**) R_M = \max\{R \mid R \leq f(B_i), R \in \mathcal{V}\}$, $f(B_i)$ is defined in the BBA ABR algorithm [16].

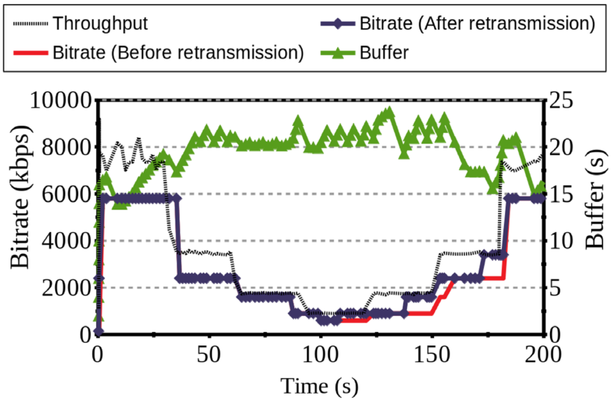


Fig. 6 An example of H2BR 's performance

thus, it is challenging to deliver this video over such an unstable network (cf. high standard deviation).

4.2 Buffer size

For AGG, we consider the buffer size of 20 s as the default value, which is similar to [20, 47]. Because the throughput-based AGG algorithm is not affected by the buffer size, we use this algorithm to assess the impact of different buffer sizes from 10 s to 20 s on the performance of H2BR. For the BBA and SARA algorithms, the buffer size should be higher since they rely on this parameter (see Table 5).

4.3 Test sequence

We encode the *RaceNight*[1] video with four different bitrate ladders as shown in Table 3 for both HEVC and SHVC. The HEVC Test Model (HM version 16.10) was used to encode the video in non-scalable format and the SHVC Test Model (SHM version 12.4[2]), which is implemented on top of HM version 16.10, was used to encode the video in the scalable format.

- *Bitrate ladder 1* conforms to YouTube[3] and includes four quality versions. Different segment durations of 1 s, 2 s, 4 s, and 6 s, and the frame rate of 30 FPS are being used.
- *Bitrate ladder 2* is based on the recommendation of Apple [1] with 12 quality versions.
- *Bitrate ladder 3* consists of 19 quality versions according to the dataset of Zabrovskiy et al. [55].
- *Bitrate ladder 4* indicates the bitrate of each layer of segments separately (from BL to EL_3). The average VMAF value of an EL is calculated when that layer is accumulated with lower layers.

Bitrate ladders 2, 3, and *4* are divided into segments with 2 s length. The first three bitrate ladders are used to investigate the impact of the bitrate ladder on the performance of H2BR.

[1] http://ultravideo.cs.tut.fi/#testsequences
[2] https://hevc.hhi.fraunhofer.de/svn/svn_SHVCSoftware/tags/SHM-12.4/
[3] https://support.google.com/youtube/answer/2853702?hl=en

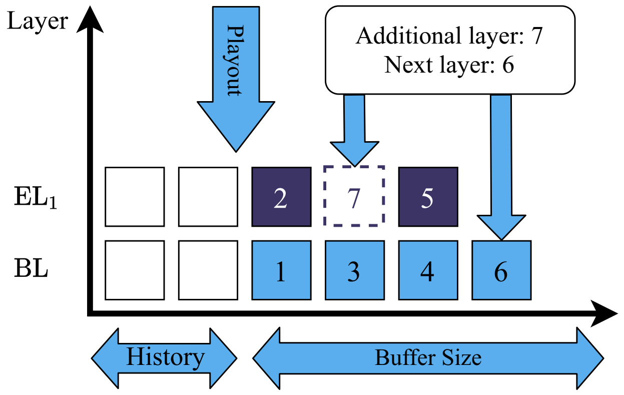


Fig. 7 H2BR in modified AGG algorithm for SVC-based HAS

Bitrate ladder 4 of SHVC will be used to compare to HEVC *Bitrate ladder 1* because the client downloads the same amount of data for the same quality version. For example, to have a 2 s segment at the quality version 2 (Fig. 1), the client has to fetch from the server one 5400 kbit/s segment in case of the HEVC video, or request the layers BL and EL_1 with the bitrates of 1800 kbit/s and 3600 kbit/s, respectively, in case of the SHVC video. The streaming session in our experiment is 300 s long. Since the *RaceNight* video is too short (12 s), it is repeatedly downloaded until the end of each session like in [47]. We used bitrate as a representative metric to evaluate the quality of segments. Because a fixed bitrate ladder is used, i.e., we encode all videos into a fixed set of representations, the performance will be similar across different content types [7].

4.4 Testbed

Our testbed consists of an HTTP/2 server running Ubuntu 18.04 LTS on an Intel Core i7 16GB RAM system and an HTTP/2 client running Ubuntu 14.04 LTS on an 8GB RAM virtual machine which are connected to each other by the *host-only* network. The *DummyNet* tool [40] is deployed at the client to emulate the network conditions mentioned

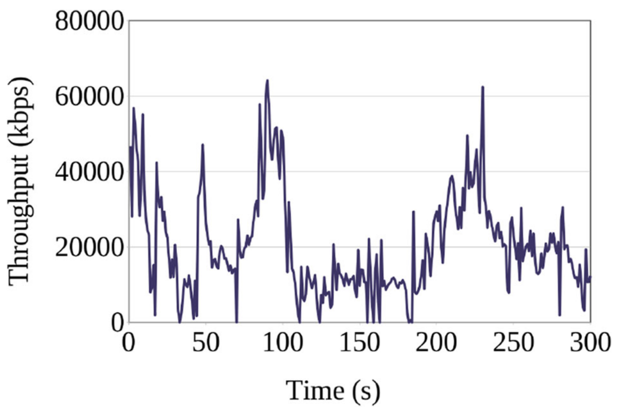


Fig. 8 Bandwidth trace [48]

Table 3 Video bitrate ladders

Video format	Ladder	Bitrate (kbps)	Average VMAF	Resolution	Video format	Ladder	Bitrate (kbps)	Average VMAF	Resolution	Video format	Ladder	Bitrate (kbps)	Average VMAF	Resolution
HEVC	1	1800	70.2	720x480	HEVC	2	8100	97.0	2560x1440	HEVC	3	4300	91.9	1920x1080
		5400	89.1	1280x720			11600	97.6	3840x2160			5800	94.4	1920x1080
		9000	97.2	1920x1080			16800	98.9	3840x2160			6500	95.6	2560x1440
		18000	99.2	3840x2160		3	100	17.6	256x144			7000	96.1	2560x1440
	2	145	40.9	640x360			200	24.4	320x180			7500	96.6	2560x1440
		300	50.1	768x432			240	30.3	384x216			8000	95.7	3840x2160
		600	61.8	960x540			375	32.0	384x216			12000	97.8	3840x2160
		900	67.0	960x540			550	44.7	512x288			17000	99.0	3840x2160
		1600	74.0	960x540			750	55.1	640x360			20000	99.3	3840x2160
		2400	83.2	1280x720			1000	63.6	768x432	SHVC	4	1800	70.4	720x480
		3400	87.2	1280x720			1500	74.5	1024x576			3600	87.9	1280x720
		4500	92.3	1920x1080			2300	82.6	1280x720			3600	94.3	1920x1080
		5800	94.4	1920x1080			3000	85.9	1280x720			9000	98.3	3840x2160

Table 4 Compared methods and acronyms

Extension	ABR algorithms				
	HEVC video streaming			SHVC video streaming	
	Throughput-based	Buffer-based	Hybrid	Throughput-based	Backfilling
NONE	AGG-N	BBA-N	SARA-N	AGG-M	BF
H2BR	AGG-H	BBA-H	SARA-H	AGG-H	–
SQUAD	AGG-S	BBA-S	SARA-S	–	–

in Section 4.1. At the client, three different ABR algorithms are implemented for HEVC video streaming: *(i)* throughput-based *AGG* [28], *(ii)* buffer-based *BBA* [16], and *(iii)* hybrid *SARA* [18]. The *k-Push* technique [28] is implemented to decrease the number of requests issued by the ABR algorithms. We compare the performance of these algorithms with and without the extensions provided by H2BR and by SQUAD's retransmission technique [49]. In the context of streaming SHVC videos, the modified AGG (*AGG-M*) approach described in Section 3 and the *Backfilling* approach [37] are deployed. These algorithms are compared to our proposed method, which is AGG-M integrated with the H2BR technique, namely *AGG-H*. The compared methods and their acronyms are listed in Table 4.

Three evaluation scenarios consisting of nine tests are considered in this article (see Table 5): (S_1) The first scenario focuses on the impact of segment duration which is in the range of 1 s to 6 s. (S_2) The second scenario considers the ability of the protocol to deal with head-of-line blocking [44] in the presence of packet loss which is set to a range from 0% to 5% based on [42]. The 2 s segment duration is used in this scenario since this is widely used in the literature on HTTP/2-based streaming [21, 27, 53]. (S_3) In the third scenario, the impact of the buffer size is investigated with the segment duration of 2 s and without packet losses. Each test is repeated ten times for accuracy, and the figures in the next sections show the average values.

Please note that *DummyNet* can only approximate the behavior of a real network. Most of these approximations are due to the limited granularity and the precision of the system clock [40]. Because of this slightly stochastic network behavior, each experiment is repeated ten times for accuracy, and the figures in the next section show the average values.

4.5 Performance metrics

To evaluate the performance of the compared methods, we consider the common metrics that characterize the QoE. They are listed in Table 6.

It should be noted that for *number of switches*, only downward switches are considered since they have a negative effect on the client's experience than upward switches [11]. For the HEVC video, we consider the QoE score computed by the extension[4] to the ITU-T P.1203 Model [17] implementation [38, 41]. The received QoE score is the result of a mapping function for HEVC video streaming that supports bitrates up to 40000 kbit/s, and video resolutions up to 4K with Root Mean Square Error (RMSE) < 0.034. Although ITU-T P.1204 originally supports HEVC and 4K videos, it was validated for video sequences of

[4]https://github.com/Telecommunication-Telemedia-Assessment/itu-p1203-codecextension, Accessed 14 June 2021.

Table 5 Experimental scenarios and configurations while using *Bitrate ladder 1*

Scenarios	Tests	Parameters				
		Segment Duration (s)	Buffer Size (s)			Packet Loss Rate (%)
			AGG	BBA	SARA	
S_1	T_1	**1**	20	44	34	0
	T_2	**2**	20	44	34	0
	T_3	**4**	20	44	34	0
	T_4	**6**	20	44	34	0
S_2	T_5	2	20	44	34	**1**
	T_6	2	20	44	34	**3**
	T_7	2	20	44	34	**5**
S_3	T_8	2	**15**	–	–	0
	T_9	2	**10**	–	–	0

Bold entries signify the best performance

short durations, e.g., 8–10s. We use the extension of the ITU-T P.1203 model, which was validated not only for supporting HEVC and 4K videos but also for videos with 1–5 minutes duration [17], which is the most suitable QoE model for our streaming scenarios.

5 Analysis: H2BR's performance in different configurations

In this section, we analyse the experimental results of H2BR in different configurations of HAS shown in Table 4. Both scalable and non-scalable video streaming are considered.

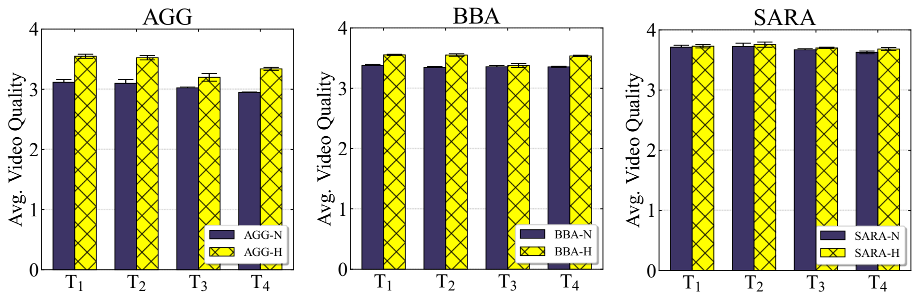
5.1 Non-scalable video streaming

First, we analyze H2BR's performance in the context of non-scalable video streaming. The results of these experiments are as follows:

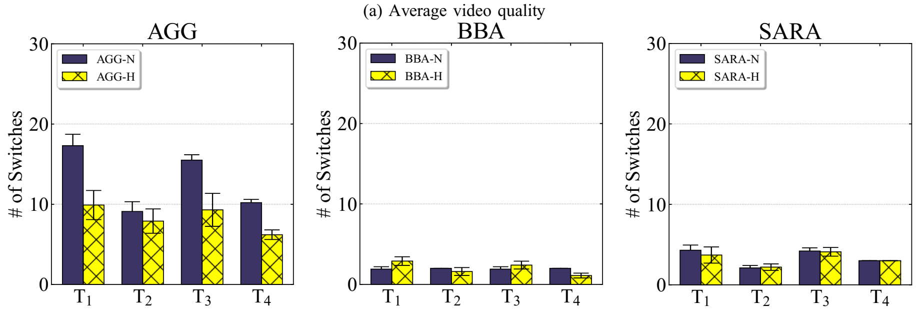
– *Impact of ABR algorithms*: H2BR is able to provide the most improvement when the throughput-based AGG algorithm is deployed at the client. The average video quality is increased by an average of 11.7% in Scenario 1 (Fig. 9(a)) and 8.7% in Scenario 2 (Fig. 10(a)). When the SARA algorithm is used, these improvements are only 0.9% and 0.3% in Scenario 1 and 2, respectively. As the AGG's buffer is well filled, H2BR is able to re-download segments successfully so that the gaps are eliminated. Up to 70 segments (out of 300 segments of the video) in T_1 as shown in Fig. 9(c) are replaced by higher qualities, which results in a decrease by an average of 36.0% in the number of switches as shown in Fig. 9(b)(left plot). Therefore, the ITU QoE score is improved the most for the AGG algorithm by an average of 10.3% and 4.9% in Scenarios 1 and 2, respectively. With the hybrid ABR algorithm, SARA, H2BR can only re-download a few segments such that SARA's performance is not significantly improved. This is because SARA only decreases the bitrate of the next segments when both the buffer and throughput are relatively low; then, the H2BR cannot be triggered. Regarding BBA, H2BR provides modest improvements. Even though this ABR algorithm relies on the buffer level, H2BR may detect enough throughput to deliver retransmitted segments

Table 6 Performance metrics

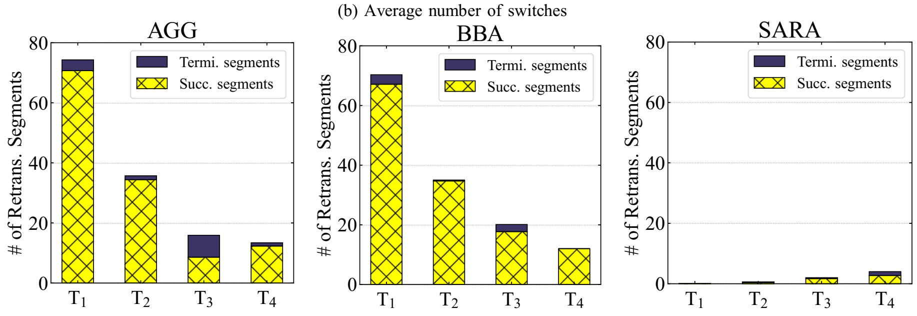
Performance metrics	Definition	
	HEVC videos	SHVC videos
Average video quality	Average quality version of segments	Average number of layers of segments
Average video instability	Average change in the quality version of adjacent segments	Average change in the number of layers in adjacent segments
Number of retransmitted segments	Number of retransmitted segments successfully downloaded	N/A (the client requests layers rather than segments)
Number of additional layers	N/A (the client requests segments rather than layers)	Number of additional layers successfully downloaded
Wasted data	Amount of data of segments replaced by higher-quality ones and not played	N/A (additional layers are decoded together with lower layers)
QoE score	Overall QoE score computed by ITU-T Rec. P.1203's extension [38, 41]	N/A (ITU-T Rec. P.1203 does not support SHVC videos)
VMAF	Video Multi-Method Assessment Fusion metric developed by Netflix [22]	
Number of switches	Number of segments which have lower quality than their previous one	
Number of stalls and Stall duration	Number of buffer underruns and total video freeze time, respectively	



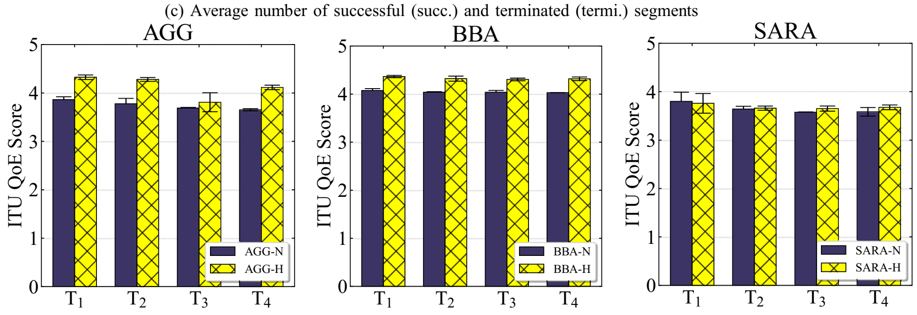
(a) Average video quality



(b) Average number of switches



(c) Average number of successful (succ.) and terminated (termi.) segments



(d) Average ITU QoE score

Fig. 9 Performance of H2BR in non-scalable video streaming over HTTP/2 for Scenario 1

when the buffer is low (but larger than B^t), the next bitrate R^N is also low, and the estimated throughput is favorable. From the above observation, we recommend that H2BR should be implemented on the clients using throughput-based ABR algorithms in all cases. Other ABR schemes can make use of H2BR in some specific scenarios that will be shown in later sections.

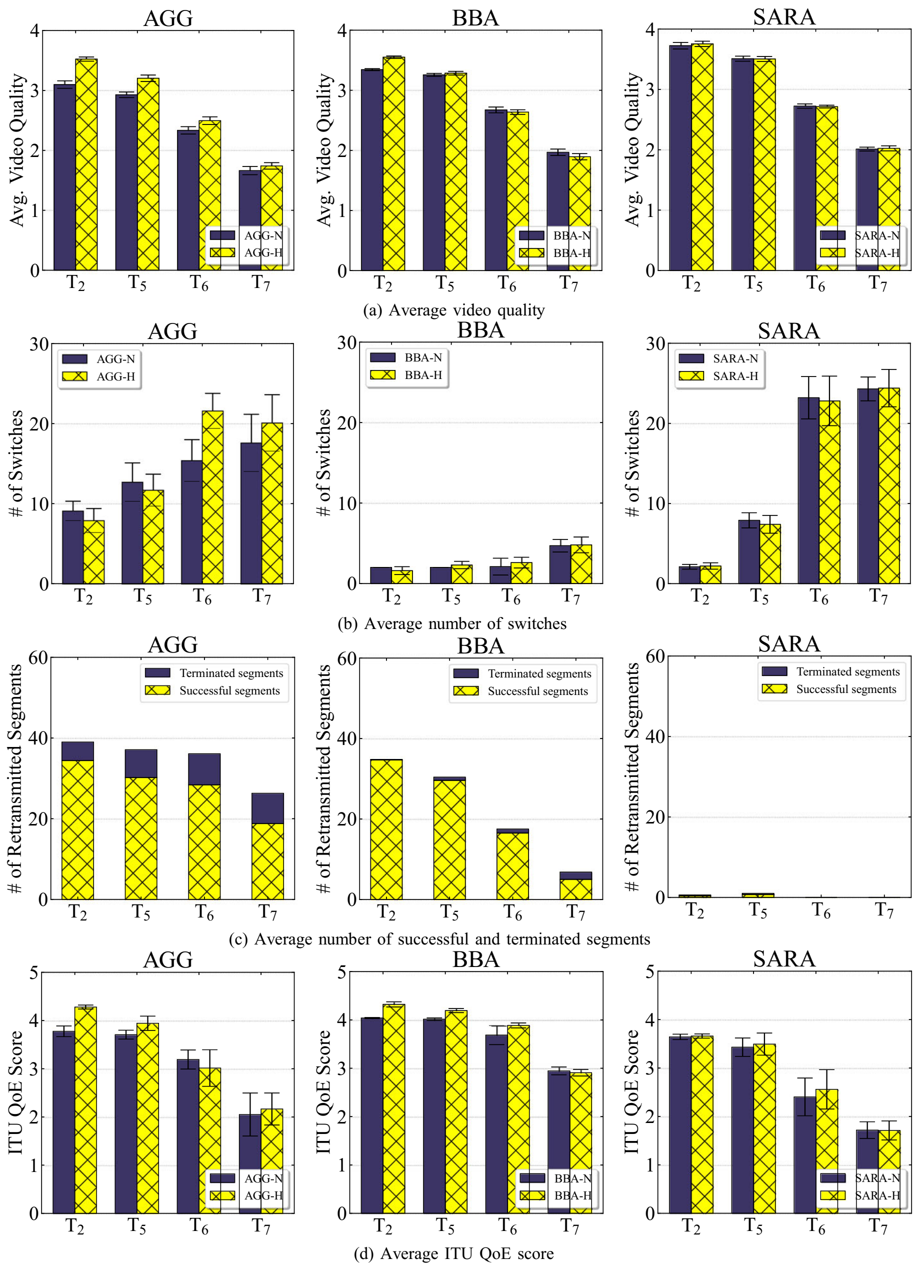


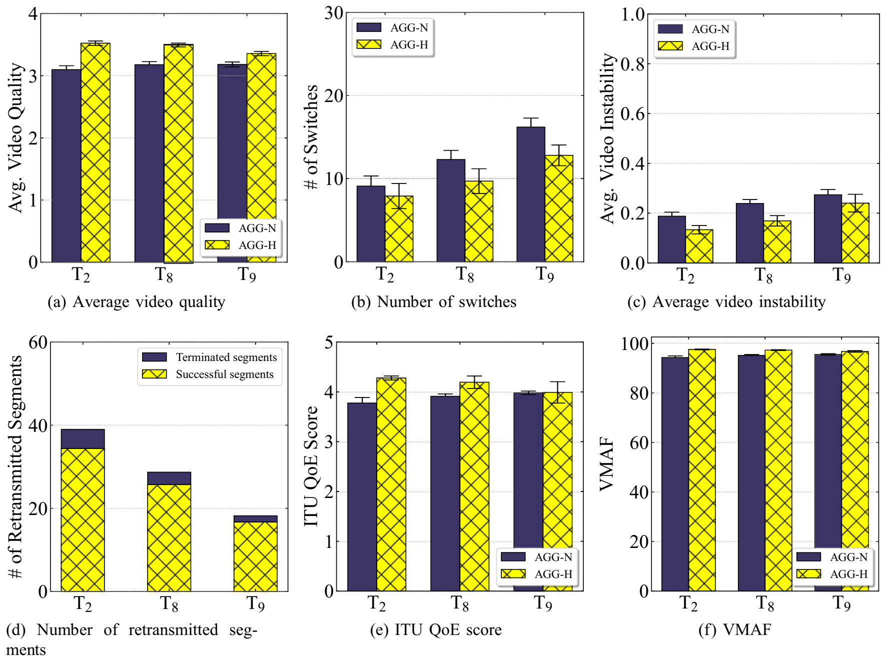
Fig. 10 Performance of H2BR in non-scalable video streaming over HTTP/2 for Scenario 2

– *Impact of segment lengths*: In Fig. 9, it can be seen that H2BR provides the best performance in T_2 (2 s segments) with an improvement in average video quality by 6.9%, the number of switches by 9.5%, and the QoE score by 6.9% for all of the ABR algorithms. Figure 9(c) shows that longer video segments lead to significantly fewer successful segment retransmissions for the AGG and BBA algorithms, but slightly affect this metric for SARA. For example, in the T_1 experiment, H2BR is able to successfully re-download 70 1-second segments on average in the AGG client, whereas this figure is only 12 6-second segments in the T_4 experiment. However, there are around 70 seconds of the video whose qualities are upgraded in all experiments except T_3. In the SARA client, no matter how long the segment duration is, no more than four segments are retransmitted completely. As explained before, when there are low-quality segments in the buffer, the buffer level and the throughput are also too small for H2BR to retransmit these segments. We can see that HTTP/2 features are exploited profitably when the video segments have 2 s segment duration.

– *Impact of packet loss rates*: H2BR is negatively affected by packet losses as shown in Fig. 10. When the packet loss rate increases, the average video quality, the number of successful segments, and the ITU QoE score deteriorate; there are more switches as well. This is because when there is a lost packet in a stream, the data transfered in its parallel streams is stopped for the TCP transport layer to recover. Therefore, the measured throughput is lower and fluctuates more severely, which results in worse bitrates and more switches at the client. As a result, H2BR cannot re-download segments or decides to re-download segments while the measured throughput is small, which leads to more terminated segments (Fig. 10(c)). Especially, H2BR cannot re-download any segments for the SARA algorithm if the packet loss rate is higher than 1%. We recommend that H2BR should be used when the network has a favorable packet loss rate such as less than 1% to avoid downloading additional data without QoE improvement.

– *Impact of buffer sizes*: As shown in Fig. 11, the larger the buffer size, the higher the improvement provided by H2BR. When the buffer size is large (20 s, i.e., T_2), H2BR achieves the best results (highest ITU QoE score) with an improvement in average video quality by 14%, the number of switches by 13%, average video instability by 29%, and ITU QoE score by 13%. This is because, with larger buffer size, there are more segments stored in the buffer, which increases the opportunity for H2BR to detect the quality gaps and retransmit low-quality segments with more available download time. It can be inferred that the client should provide enough buffer capacity (e.g., 20 s for AGG) for the purpose of achieving the best benefits of HTTP/2 features with H2BR.

– *Impact of bitrate ladders:* From Fig. 12, we observe that, with more quality versions of the video, the improvements provided by H2BR decrease in the AGG and BBA clients. For example, with *Bitrate ladder 1* (i.e., four quality versions), the average video quality is increased by 14% in AGG and 6% in BBA. For *Bitrate ladder 3*, this figure is improved by only 7% in AGG and only by 1% in BBA. The reason is that more quality versions mean the AGG approach can choose higher qualities. If the segments with these qualities need to be re-downloaded, H2BR needs more throughput; this decreases the chance of triggering retransmission and fewer segments are re-downloaded (Fig. 12(f)). Regarding the BBA algorithm, this buffer-based method uses a function to map the buffer level to the quality version. Therefore, if the video has few quality versions (i.e., *Bitrate ladder 1*), a large range of buffer levels maps to one quality version. When H2BR finishes retransmitting some segments, it does not affect the bitrate selection of this ABR algorithm despite a lower buffer occupancy. Consequently, H2BR is able to eliminate more downward switches, and the video quality is more stable when the video has fewer quality versions (Fig. 12(b) and (c)). In contrast, H2BR enhances SARA's performance if the video has many quality versions. For



(a) Average video quality (b) Number of switches (c) Average video instability (d) Number of retransmitted segments (e) ITU QoE score (f) VMAF

Fig. 11 Performance of H2BR in non-scalable video streaming over HTTP/2 for Scenario 3

Bitrate ladder 3 (i.e., 19 quality versions), H2BR improves by 4% in the average video quality, 3% in both the QoE score and VMAF, and 18% in video instability because of more retransmitted segments, as shown in Fig. 12(f). This is because when the buffer is high enough to trigger retransmission, SARA also wants to download the next segments with high quality, which leaves little throughput for H2BR. Therefore, if the video has more quality versions, H2BR has more chance of finding a suitable low bitrate that fits with that remaining throughput. We conclude that buffer-based ABR algorithms like BBA can use H2BR when the clients download video with a few quality levels, whereas hybrid schemes like SARA should use H2BR when the video is encoded into many quality versions.

5.2 Scalable video streaming

Figures 13, 14, and 15 compare the performance of our proposed method with that of the modified AGG algorithm, AGG-M, in scalable video streaming with different scenarios. The following results are obtained.

– *Impact of segment lengths:* Fig. 13 shows that our proposed method AGG-H outperforms AGG-M for each segment duration with an improvement in the average video quality of 15.4%, the number of switches of 81%, and the average video instability of 64.8%. However, it is seen that longer segments decrease gradually the performance gain of AGG-H when compared to AGG-M. For example, the average video quality rises by 23% (from 3.0 to 3.7) in T_1 (i.e., 1 s segments) whereas this gain is only 10% in T_4 (i.e., 6 s segments). This is because with longer segments, the video has

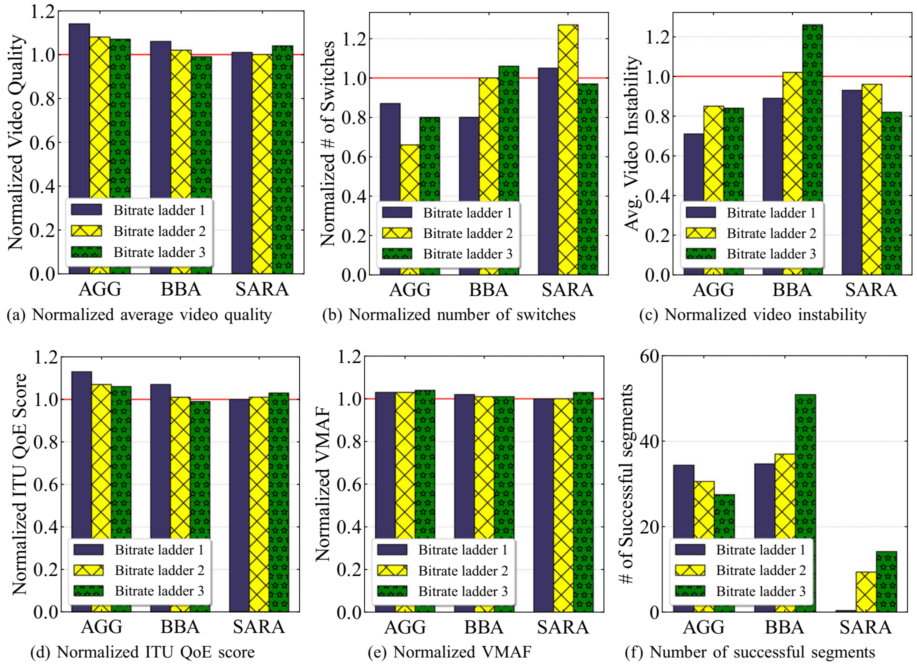


Fig. 12 Performance of H2BR in non-scalable video streaming with different video ladders in T_2

fewer video segments, and the client more slowly adapts to the fluctuation of the network but might also cause the switches to become more severe. Consequently, there are fewer downward switches (Fig. 13(b)), and higher average video instability (Fig. 13(c)) in AGG-M. For AGG-H, as explained before, longer segments mean the requirement of more download time that decreases the chance of additionally downloading layers. Therefore, fewer downward switches are eliminated and also more average video instability emerges. From these results, the segment duration in the scalable video needs to be short (e.g., 1-2seconds) to achieve the best quality from H2BR.

– *Impact of packet loss rates:* As illustrated in Fig. 14, packet losses, evidently, negatively affect the performance of our proposed method. The higher the packet loss rate,

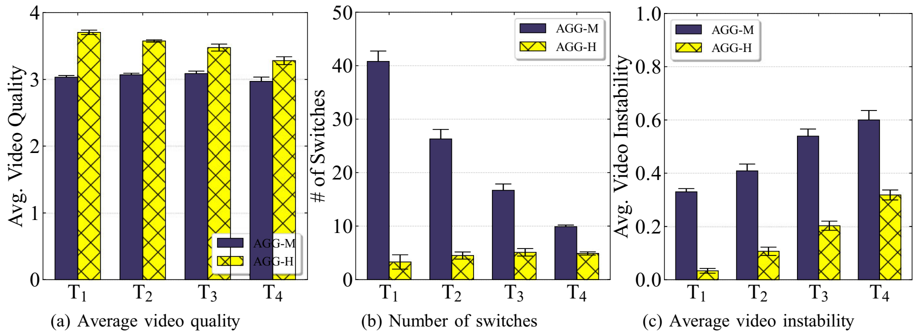


Fig. 13 Performance of approaches in scalable video streaming for Scenario 1

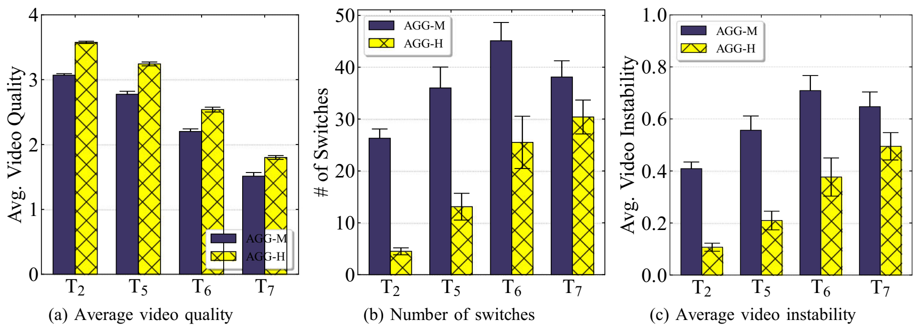


Fig. 14 Performance of approaches in scalable video streaming for Scenario 2

the smaller the performance gain yielded by H2BR. In T_2 (i.e., a packet loss rate of 0%), AGG-H is able to bring 16%, 83%, and 74% improvements in average video quality, number of switches, and average video instability, respectively. The corresponding gains in T_7 (i.e., a packet loss rate of 5%) are 17%, 20%, and 19%. As explained above, higher packet loss rates lead to more throughput oscillation and reduction, which challenges AGG-H to improve over AGG-M's results. Regarding the decrease in the number of switches and video instability of AGG-M in T_7 when compared to T_6, these are attributed to the serious reduction of the throughput such that the chosen video quality is between 1 and 2 (the average video quality is 1.5 as shown in Fig. 14(a)). However, it can be clearly seen that H2BR utilizes HTTP/2 features efficiently for scalable video streaming even when the network has a high packet loss rate (e.g., 5%) with an average quality improvement of 17%.

– *Impact of buffer sizes:* Fig. 15 shows the performance of the compared methods in the tests T_2, T_8, and T_9 with buffer sizes of 20 s, 15 s, and 10 s, respectively. Similar to non-scalable video streaming in the previous section, shorter buffer sizes result in more switches and higher average video instability in AGG-H. However, our proposed method still outperforms AGG-M.

5.3 Guidelines

In this section, we have evaluated H2BR in various streaming scenarios. The main findings and guidelines on H2BR can be drawn as follows: *(i)* If the client implements a

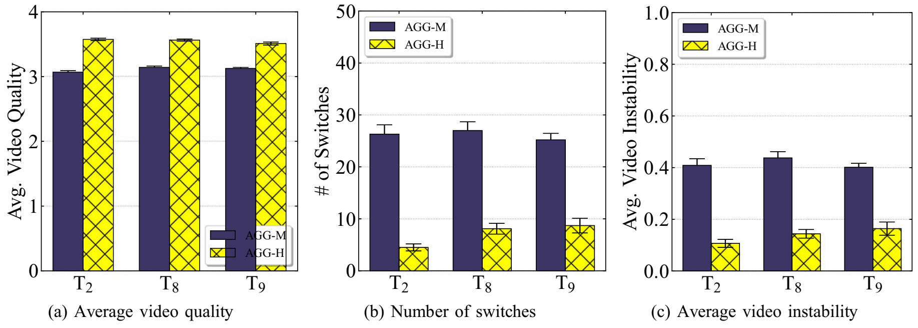


Fig. 15 Performance of approaches in scalable video streaming for Scenario 3

throughput-based ABR algorithm, H2BR should be integrated for every streaming scenario to enhance the QoE. *(ii)* We recommend applying H2BR for buffer-based ABR schemes only when the videos have few quality levels. *(iii)* For hybrid ABR methods, H2BR is profitable if the videos are encoded at many quality levels. *(iv)* For the purpose of exploiting HTTP/2 features effectively, the segment duration should be 2s, and the clients should allocate large buffers. *(v)* H2BR gains remarkable QoE improvement in the context of scalable video streaming in all considered scenarios.

6 Analysis: H2BR's performance compared to related work

In this section, we compare H2BR with state-of-the-art approaches in scalable and non-scalable video streaming.

6.1 Non-scalable video streaming

In the context of non-scalable video streaming, Table 7 compares the performance of H2BR and the SQUAD method [49] with different bitrate ladders and in combination with different ABR algorithms in test T_2. The main observation is that H2BR provides better results than SQUAD in almost all cases.

For the AGG algorithm, our proposed method (i.e., AGG-H) provides the best performance in all three bitrate ladders. For instance, with *Bitrate ladder 1*, AGG-H achieves an average video quality of 3.5, VMAF of 97.5, and an average QoE score of 4.3, compared to 3.2, 95.3, and 3.9, respectively, of AGG-S, and 3.1, 94.2, and 3.8 of AGG-N. This is attributed to much more successful segment retransmissions with higher qualities by H2BR. There is an average of 34.3 segments successfully re-downloaded by the proposed method, compared to 14.1 segments by SQUAD for *Bitrate ladder 1*. Moreover, as AGG is a throughput-based method, retransmissions do not much affect the bitrate decisions on the future segments of AGG; thus, more successful segments mean higher overall quality. Nevertheless, H2BR provides these better results at the cost of more wasted data (nearly 46 MB, more than 2.5 times of SQUAD). Although the number of switches in H2BR is higher than that of SQUAD, it still appears reasonable with a decrease by 13% to 34% in the number of switches and 15% to 29% in the video instability when compared to AGG-N in all cases of bitrate ladders.

With the BBA client, both H2BR (i.e., BBA-H) and SQUAD (i.e., BBA-S) retransmission techniques cannot remarkably improve the performance of the original ABR algorithm (i.e., BBA-N) when the video is encoded with many quality versions (i.e., *Bitrate ladders 2* and *3*). Although multiple segments are retransmitted (up to 60 segments in H2BR and 49 segments in SQUAD with *Bitrate ladder 3*), the results are relatively unchanged. As explained before, this is attributed to the negative impact of retransmission on the buffer occupancy, which is used for the future bitrate selection of BBA. This issue abates in *Bitrate ladder 1* where the changes in the buffer level because of retransmissions hardly distort the chosen bitrates of the original BBA. Therefore, BBA-H achieves the best results with the average video quality of 3.6 (6% higher than the others), the average QoE score of 4.3 (nearly 5% higher than BBA-S and 8% higher than BBA-N), and the average number of switches of 1.6, compared to 1.8 and 2.0 of BBA-S and BBA-N, respectively. We conclude that the retransmission technique should not be used in buffer-based ABR schemes while delivering videos that have many (e.g., 12) quality versions.

Table 7 Performance metrics for different bitrate ladders in the T2 experiment

	Avg. Video Quality	Avg. # of Switches	Avg. Instability	Avg. # of Successful Segments	Avg. Wasted Data (MBytes)	Avg. # of Stalls & Duration (s)	Avg. VMAF	Avg. QoE
	Bitrate ladder 1 (4 representations)							
AGG-N	3.0 to 3.2 (3.1)	7 to 11 (9.1)	0.16 to 0.21 (0.19)	–	–	0 &0	93.3 to 95.2 (94.2)	3.6 to 4.0 (3.8)
AGG-H	**3.5 to 3.6 (3.5)**	**6 to 11 (7.9)**	**0.11 to 0.16 (0.13)**	**23 to 44 (34.3)**	**28.4 to 56.8 (45.9)**	**0 & 0**	**97.2 to 97.7 (97.5)**	**4.2 to 4.3 (4.3)**
AGG-S	3.1 to 3.3 (3.2)	5 to 9 (6.7)	0.11 to 0.19 (0.14)	8 to 20 (14.1)	9.0 to 25.7 (16.5)	0 & 0	93.2 to 96.1 (95.3)	3.3 to 4.1 (3.9)
BBA-N	3.3 to 3.4 (3.3)	2 to 2 (2)	0.05 to 0.05 (0.05)	–	–	0 & 0	95.9 to 96.2 (96.0)	4.0 to 4.1 (4.0)
BBA-H	**3.5 to 3.6 (3.6)**	**1 to 2 (1.6)**	**0.03 to 0.05 (0.04)**	**29 to 41 (34.7)**	**47.3 to 66.3 (54.50)**	**0 & 0**	**97.6 to 98.0 (97.8)**	**4.2 to 4.4 (4.3)**
BBA-S	3.3 to 3.4 (3.3)	1 to 2 (1.8)	0.03 to 0.05 (0.04)	0 to 10 (1.8)	0 to 13.5 (2.4)	0 & 0	95.9 to 96.3 (96.0)	4.0 to 4.1 (4.1)
SARA-N	3.62 to 3.79 (3.7)	2 to 3 (2.1)	0.07 to 0.13 (0.10)	–	–	2.1 & 9.3	96.4 to 97.8 (97.2)	3.5 to 3.7 (3.6)
SARA-H	**3.66 to 3.79 (3.8)**	**2 to 3 (2.2)**	**0.07 to 0.11 (0.09)**	**0 to 2 (0.4)**	**0 to 2.7 (0.5)**	**2.2 & 10.4**	**97.0 to 97.8 (97.5)**	**3.6 to 3.7 (3.7)**
SARA-S	3.69 to 3.79 (3.7)	1 to 3 (2.1)	0.06 to 0.13 (0.10)	0 to 0 (0)	0 to 0 (0)	.2.2 & 8.8	96.9 to 97.9 (97.3)	3.6 to 3.7 (3.6)
	Bitrate ladder 2 (12 representations)							
AGG-N	10.3 to 10.7 (10.5)	9 to 14 (12.2)	0.27 to 0.45 (0.37)	–	–	0 & 0	93.7 to 94.9 (94.4)	4.0 to 4.2 (4.2)
AGG-H	**11.3 to 11.4 (11.3)**	**6 to 10 (8.1)**	**0.17 to 0.44 (0.31)**	**22 to 36 (30.6)**	**33.0 to 58.1 (47.9)**	**0 & 0**	**97.3 to 97.9 (97.6)**	**4.4 to 4.5 (4.5)**
AGG-S	10.7 to 10.9 (10.7)	6 to 11 (8.3)	0.21 to 0.44 (0.35)	9 to 22 (15.0)	13.5 to 33.5 (23.7)	0 & 0	94.3 to 95.0 (94.7)	4.2 to 4.3 (4.2)
BBA-N	10.7 to 10.7 (10.7)	2 to 3 (2.5)	0.11 to 0.13 (0.12)	–	–	0 & 0	95.0 to 95.2 (95.1)	4.2 to 4.3 (4.2)
BBA-H	**10.9 to 11.0 (10.9)**	**2 to 4 (2.5)**	**0.10 to 0.14 (0.12)**	**30 to 50 (37)**	**52.5 to 93.1 (66.1)**	**0 & 0**	**96.1 to 96.5 (96.3)**	**4.3 to 4.4 (4.3)**
BBA-S	10.6 to 10.7 (10.7)	2 to 3 (2.7)	0.10 to 0.11 (0.11)	0 to 21 (9.2)	0 to 28.0 (12.8)	0 & 0	95.0 to 95.3 (95.2)	4. 2 to 4.3 (4.2)
SARA-N	10.6 to 10.8 (10.8)	1 to 2 (1.1)	0.22 to 0.28 (0.23)	–	–	1.6 & 2.4	92.7 to 94.2 (93.4)	3.5 to 4.0 (3.6)
SARA-H	**10.7 to 10.9 (10.8)**	**1 to 2 (1.4)**	**0.21 to 0.22 (0.22)**	**8 to 11 (9.4)**	**0.8 to 1.4 (1.0)**	**1.4 & 1.9**	**93.5 to 94.1 (93.8)**	**3.5 to 3.9 (3.6)**
SARA-S	10.6 to 10.7 (10.7)	1 to 2 (1.2)	0.22 to 0.32 (0.23)	0 to 1 (0.1)	0 to 0.04 (0)	1.6 & 2.4	92.9 to 93.7 (93.4)	3.5 to 3.9 (3.6)

Table 7 (continued)

	Avg. Video Quality	Avg. # of Switches	Avg. Instability	Avg. # of Successful Segments	Avg. Wasted Data (MBytes)	Avg. # of Stalls & Duration (s)	Avg. VMAF	Avg. QoE
	Bitrate ladder 3 (19 representations)							
AGG-N	16.0 to 16.6 (16.3)	10 to 16 (13.4)	0.44 to 0.73 (0.62)	–	–	0 & 0	92.6 to 93.5 (93.1)	4.1 to 4.3 (4.2)
AGG-H	**17.3 to 17.6 (17.5)**	**7 to 15 (10.7)**	**0.38 to 0.66 (0.52)**	**23 to 35 (27.5)**	**46.5 to 80.3 (56.7)**	**0 & 0**	**96.4 to 97.1 (96.8)**	**4.4 to 4.5 (4.5)**
AGG-S	16.5 to 16.8 (16.7)	7 to 11 (9.6)	0.32 to 0.58 (0.44)	13 to 21 (16.5)	22.1 to 45.3 (31.4)	0 & 0	92.4 to 93.7 (93.4)	4.1 to 4.3 (4.3)
BBA-N	**16.6 to 16.9 (16.7)**	**5 to 6 (5.2)**	**0.21 to 0.30 (0.28)**	**–**	**–**	**0 & 0**	**94.3 to 94.5 (94.4)**	**4.2 to 4.3 (4.3)**
BBA-H	16.4 to 16.7 (16.6)	4 to 7 (5.9)	0.30 to 0.40 (0.34)	46 to 60 (50.9)	105.3 to 122.1 (115.1)	0 & 0	95.5 to 95.9 (95.7)	4.2 to 4.3 (4.3)
BBA-S	16.0 to 16.3 (16.1)	2 to 5 (3.3)	0.23 to 0.36 (0.28)	26 to 49 (34.0)	66.5 to 90.3 (76.9)	0 & 0	92.8 to 94.0 (93.6)	4.1 to 4.2 (4.2)
SARA-N	15.63 to 16.85 (15.86)	3 to 4 (3.1)	0.36 to 0.63 (0.59)	–	–	1 & 0.1	85.2 to 93.7 (86.7)	3.2 to 4.3 (3.4)
SARA-H	**15.96 to 16.93 (16.51)**	**2 to 4 (3.0)**	**0.35 to 0.63 (0.49)**	**9 to 22 (14.2)**	**0.7 to 19.3 (3.6)**	**0.9 & 2.5**	**86.8 to 93.8 (89.7)**	**3.3 to 4.1 (3.5)**
SARA-S	15.63 to 16.80 (15.95)	3 to 5 (3.3)	0.39 to 0.62 (0.56)	0 to 10 (2.3)	0.0 to 0.4 (0.1)	1.8 & 1.7	85.2 to 90.6 (86.8)	3.2 to 3.5 (3.3)

Bold entries signify the best performance

Regarding the SARA algorithm, H2BR (i.e., SARA-H) can yield the best performance in all bitrate ladders, and the more quality versions the video has, the better performance H2BR can achieve. With *Bitrate ladder 1*, the results of using retransmission and not using retransmission are nearly the same. SQUAD (i.e., SARA-S) could not retransmit any segments, and H2BR can successfully download a maximum of two segments to replace the low-quality ones. However, when the video has more quality versions (e.g., *Bitrate ladder 3*), H2BR shows the best improvement over the original SARA (i.e., SARA-N) owing to the average of about 14 segments completely retransmitted and 3.6 MB video data replaced, whereas SQUAD could hardly re-download segments. The average video quality and average video instability of our proposed method are 16.5 and 0.49, respectively, compared to 15.9 and 0.59 of SARA-N. Although SARA-H suffers from a longer stall duration (2.5 s), its average VMAF is significantly higher at 89.7, compared to 86.7 of SARA-N.

6.2 Scalable video streaming

When the videos are encoded with the scalable format, Table 8 summarizes the performance of our proposed method AGG-H and two reference approaches (AGG-M and BF). AGG-M is the AGG algorithm modified to be compatible with the SVC streaming scenario, and BF stands for the Backfilling approach [37]. Meanwhile, AGG-H is the modified AGG integrated with the H2BR technique. It can be seen that AGG-H provides the best video quality. Owing to an average of 63.6 layers additionally downloaded, AGG-H enhances the performance of AGG-M by 14% in average video quality, 60% in average number of switches, 60% in average video instability, and 3% in average VMAF, with a reasonable number of stalls and stall duration. Moreover, compared to BF, our proposed method is able to achieve 39% fewer switches on average (from 19.3 down to 11.8), and 50% less average video instability (from 0.4 down to 0.2) with a slightly higher average video quality and average VMAF.

7 Conclusions and future work

H2BR is a retransmission technique that leverages HTTP/2 features to enhance the QoE. The proposed H2BR approach scans the buffer to detect lower-quality segments compared to the adjacent ones and determines higher-quality versions based on the buffer occupancy and the estimated throughput. Our proposed approach makes the retransmitted segments arrive at the client on time and diminish the request overhead by leveraging the following HTTP/2 features on top of existing ABR algorithms: *server push*, *stream priority*, *multiplexing*, and *stream termination*. These features enable H2BR to upgrade the quality of segments which are worse than their neighboring segments while avoiding adversely affecting the performance of the underlying ABR algorithms. We intensely analyze the performance of H2BR in different scenarios, varying parameters related to the network (i.e., packet loss rate), the client (i.e., buffer size and ABR algorithm), and the video (i.e., segment duration and the number of quality versions). From the experimental results, we found that H2BR is able to improve the video quality of the existing ABR algorithms by up to 17% and 14% in scalable video streaming and non-scalable video streaming, respectively. H2BR also outperforms the retransmission technique SQUAD by more than 10% higher QoE. Furthermore, H2BR significantly enhances the quality of videos when the client integrates a throughput-based ABR method with a suitable buffer size (around 20 s), and at a favorable packet loss rate (no more than 1%). In the case of a buffer-based ABR algorithm, bitrate ladders with

Table 8 Performance of the SVC-based ABR approaches in all experiments

	Avg. Video Quality	Avg. # of Switches	Avg. Instability	Avg. # of Successful Layers	Avg. # of Stalls & Duration (s)	Avg. VMAF
AGG-M	2.8	29.5	0.5	–	0.2 & 0.4	90.1
AGG-H	**3.2**	**11.8**	**0.2**	63.6	0.4 & 1.0	**93.1**
BF	3.1	19.3	0.4	–	**0 & 0**	92.8

Bold entries signify the best performance a small number of quality versions are important for H2BR to bring benefits for the client. In contrast, our proposed method is able to improve the QoE for the hybrid ABR algorithm when the video is encoded into many quality versions. We also observe that H2BR provides more improvements for scalable video than for non-scalable video with, on average, 14% in average video quality, 62% in the number of switches, and 57% in average video instability, compared to 9%, 15%, and 13% for the non-scalable video, respectively.

We also provide recommendations on how to utilize H2BR effectively as follows. Firstly, H2BR is beneficial when using a throughput-based ABR algorithm in any scenario to achieve better QoE. Secondly, when the HAS client uses a buffer-based ABR scheme, H2BR should be used if that client is watching a video with few quality levels, such as less than 12. Thirdly, hybrid ABR methods should trigger H2BR when the video has many quality levels (e.g., 19 levels). Finally, the segment duration should be 2 s to make H2BR obtain the best performance.

Future work includes implementing H2BR in conjunction with multipath TCP (MPTCP) where the retransmitted segments and next segments can be downloaded via separate interfaces, e.g., a cellular network and Wi-Fi, to eliminate the interference of H2BR with the original ABR algorithms.

Funding Open access funding provided by University of Klagenfurt.

Acknowledgements The financial support of the Austrian Federal Ministry for Digital and Economic Affairs, the National Foundation for Research, Technology, and Development, and the Christian Doppler Research Association is gratefully acknowledged. Christian Doppler Laboratory ATHENA: https://athena.itec.aau.at/.

Data availability statement The datasets generated during and/or analysed during the current study are available from the corresponding author on reasonable request.

Declarations

Conflict of Interests No conflict to declare by the authors.

References

1. Apple (2020) HLS authoring specification for Apple devices [Online] Available: https://developer.apple.com/documentation/http_live_streaming/hls_authoring_specification_for_apple_devices#//apple_ref/doc/uid/TP40016596-CH4-SW1. Accessed 1 Nov 2020
2. AT&T Developer Adaptive bitrate video streaming [Online] Available: https://developer.att.com/video-optimizer/docs/best-practices/adaptive-bitrate-video-streaming. Accessed 06 April 2021
3. Batalla JM, Krawiec P, Beben A, Wisniewski P, Chydzinski A (2016) Adaptive video streaming: rate and buffer on the track of minimum rebuffering. IEEE J Sel Areas Commun 34(8):2154–2167. https://doi.org/10.1109/JSAC.2016.2577360
4. Belda R, de Fez I, Arce P, Guerri JC (2020) Look ahead to improve QoE in DASH streaming. Multimed Tools Applic 79(33):25143–25170. https://doi.org/10.1007/s11042-020-09214-9

5. Belshe M, Peon R, Thomson M (2015) Hypertext transfer protocol version 2 (HTTP/2). RFC 7540. https://tools.ietf.org/html/rfc7540
6. Bentaleb A, Taani B, Begen AC, Timmerer C, Zimmermann R (2018) A survey on bitrate adaptation schemes for streaming media over HTTP. IEEE Commun Surv Tutor 21(1):562–585. https://doi.org/10.1109/COMST.2018.2862938
7. Bentaleb A, Timmerer C, Begen AC, Zimmermann R (2020) Performance analysis of ACTE: a bandwidth prediction method for low-latency chunked streaming. ACM Trans Multimed Comput Commun Applic (TOMM) 16(2s):1–24. https://doi.org/10.1145/3387921
8. Bhat D, Deshmukh R, Zink M (2018) Improving QoE of ABR streaming sessions through QUIC retransmissions. In: ACM Multimedia Conf. (MM). ACM, pp 1616–1624. https://doi.org/10.1145/3240508.3240664
9. Bitmovin Inc (2020) 2020 video developer report [Online] Available: https://go.bitmovin.com/video-developer-report-2020 bitmovin.com (ed). Accessed 08 July 2021
10. Bouten N, Latré S, Famaey J, De Turck F, Van Leekwijck W (2013) Minimizing the impact of delay on live SVC-based HTTP adaptive streaming services. In: 2013 IFIP/IEEE Int'l. symposium on integrated network management (IM 2013), pp 1399–1404
11. Cranley N, Perry P, Murphy L (2006) User perception of adapting video quality. Int J Human-Comput Stud 64(8):637–647. https://doi.org/10.1016/j.ijhcs.2005.12.002
12. Dobrian F, Sekar V, Awan A, Stoica I, Joseph D, Ganjam A, Zhan J, Zhang H (2011) Understanding the impact of video quality on user engagement. In: ACM SIGCOMM computer communication review, vol 41. ACM, pp 362–373. https://doi.org/10.1145/2043164.2018478
13. Elgabli A, Aggarwal V, Hao S, Qian F, Sen S (2018) LBP: robust rate adaptation algorithm for SVC video streaming. IEEE/ACM Trans Network 26(4):1633–1645. https://doi.org/10.1109/TNET.2018.2844123
14. Grafl M, Timmerer C, Hellwagner H, Cherif W, Ksentini A (2013) Evaluation of hybrid scalable video coding for HTTP-based adaptive media streaming with high-definition content. In: 2013 IEEE 14th Int'l. symposium world of wireless, mobile and multimedia networks (WoWMoM), pp 1–7. https://doi.org/10.1109/WoWMoM.2013.6583506
15. Hoßfeld T, Seufert M, Sieber C, Zinner T (2014) Assessing effect sizes of influence factors towards a QoE model for HTTP adaptive streaming. In: Sixth Int'l. workshop on quality of multimedia experience (QoMEX). IEEE, pp 111–116. https://doi.org/10.1109/QoMEX.2014.6982305
16. Huang TY, Johari R, McKeown N, Trunnell M, Watson M (2014) A buffer-based approach to rate adaptation: evidence from a large video streaming service. In: ACM SIGCOMM computer communication review, vol 44. ACM, pp 187–198. https://doi.org/10.1145/2619239.2626296
17. ITU-T Rec. P.1203. Parametric bitstream-based quality assessment of progressive download and adaptive audiovisual streaming services over reliable transport - video quality estimation module. http://handle.itu.int/11.1002/ps/P1203-01, Accessed 08 July 2021
18. Juluri P, Tamarapalli V, Medhi D (2015) SARA: segment aware rate adaptation algorithm for dynamic adaptive streaming over HTTP. In: IEEE Int'l. conf. on communication workshops (ICCW). IEEE, pp 1765–1770. https://doi.org/10.1109/ICCW.2015.7247436
19. Kalva H, Adzic V, Furht B (2012) Comparing MPEG AVC and SVC for adaptive HTTP streaming. In: 2012 IEEE international conference on consumer electronics (ICCE). IEEE, pp 158–159. https://doi.org/10.1109/ICCE.2012.6161787
20. Le HT, Nguyen DV, Pham NN, Pham AT, Thang TC (2013) Buffer-based bitrate adaptation for adaptive HTTP streaming. In: 2013 Int'l. conf. on advanced technologies for communications (ATC 2013). IEEE, pp 33–38. https://doi.org/10.1109/ATC.2013.6698072
21. Le HT, Nguyen T, Ngoc NP, Pham AT, Thang TC (2018) HTTP/2 push-based low-delay live streaming over mobile networks with stream termination. IEEE Trans Circuits Syst Video Technol 28(9):2423–2427. https://doi.org/10.1109/TCSVT.2018.2850740
22. Li Z, Aaron A, Katsavounidis I, Moorthy A, Manohara M Toward a practical perceptual video quality metric. [Online] Available: https://netflixtechblog.com/toward-a-practical-perceptual-video-quality-metric-653f208b9652. Accessed 20 June 2021
23. Li Z, Zhu X, Gahm J, Pan R, Hu H, Begen AC, Oran D (2014) Probe and adapt: rate adaptation for HTTP video streaming at scale. IEEE J Sel Areas Commun 32(4):719–733
24. Liu C, Bouazizi I, Gabbouj M (2011) Rate adaptation for adaptive HTTP streaming. In: Proceedings of the second annual ACM conference on multimedia systems, pp 169–174. https://doi.org/10.1145/1943552.1943575
25. Liu C, Bouazizi I, Hannuksela MM, Gabbouj M (2012) Rate adaptation for dynamic adaptive streaming over HTTP in content distribution network. Signal Process: Image Commun 27(4):288–311. https://doi.org/10.1016/j.image.2011.10.001

26. Miller K, Al-Tamimi A-K, Wolisz A (2016) QoE-based low-delay live streaming using throughput predictions. ACM Trans Multimed Comput Commun Applic (TOMM) 13(1):1–24
27. Müller C, Lederer S, Timmerer C, Hellwagner H (2013) Dynamic adaptive streaming over HTTP/2.0. In: 2013 IEEE Int'l. conf. on multimedia and expo (ICME), pp 1–6. https://doi.org/10.1109/ICME.2013.6607498
28. Nguyen DV, Le HT, Nam PN, Pham AT, Thang TC (2016) Adaptation method for video streaming over HTTP/2. IEICE Commun Express 5(3):69–73. https://doi.org/10.1587/comex.2015XBL0177
29. Nguyen DH, Nguyen M, Ngoc NP, Thang TC (2018) An adaptive method for low-delay 360 VR video streaming over HTTP/2. In: Seventh IEEE Int'l. conf. on communications and electronics (ICCE). IEEE, pp 261–266. https://doi.org/10.1109/CCE.2018.8465722
30. Nguyen DV, Van Trung H, Huong HLD, Huong TT, Ngoc NP, Thang TC (2019) Scalable 360 video streaming using HTTP/2. In: 2019 IEEE 21st Int'l. workshop on multimedia signal processing (MMSP). IEEE, pp 1–6. https://doi.org/10.1109/MMSP.2019.8901805
31. Nguyen M, Nguyen DH, Pham CT, Ngoc NP, Nguyen DV, Thang TC (2017) An adaptive streaming method of 360 videos over HTTP/2 protocol. In: 4th NAFOSTED conf. on information and computer science. IEEE, pp 302–307. https://doi.org/10.1109/NAFOSTED.2017.8108082
32. Nguyen M, Amirpour H, Timmerer C, Hellwagner H (2020) Scalable high efficiency video coding based HTTP adaptive streaming over QUIC. In: Proceedings of the workshop on the evolution, performance, and interoperability of QUIC, pp 28–34. https://doi.org/10.1145/3405796.3405829
33. Nguyen M, Timmerer C, Hellwagner H (2020) H2BR: an HTTP/2-based retransmission technique to improve the QoE of adaptive video streaming. In: Proceedings of the 25th packet video workshop, pp 1–7. https://doi.org/10.1145/3386292.3397117
34. Nguyen M, Çetinkaya E, Hellwagner H, Timmerer C (2021) WISH: user-centric bitrate adaptation for HTTP adaptive streaming on mobile devices. In: 2021 IEEE 23rd international workshop on multimedia signal processing (MMSP). IEEE, pp 1–6
35. Oelbaum T, Schwarz H, Wien M, Wiegand T (2008) Subjective performance evaluation of the SVC extension of H.264/AVC. In: 2008 15th IEEE int'l. conf. on image processing, pp 2772–2775. https://doi.org/10.1109/ICIP.2008.4712369
36. Ogasawara T, Bandai M (2019) An interest control method in SVC-based adaptive streaming over named data networking. In: 2019 16th IEEE annual consumer communications networking conf. (CCNC), pp 1–4. https://doi.org/10.1109/CCNC.2019.8651792
37. Petrangeli S, Bouten N, Claeys M, De Turck F (2015) Towards SVC-based adaptive streaming in information centric networks. In: 2015 IEEE int'l. conf. on multimedia expo workshops (ICMEW), pp 1–6. https://doi.org/10.1109/ICMEW.2015.7169859
38. Raake A, Garcia MN, Robitza W, List P, Göring S, Feiten B (2017) A bitstream-based, scalable video-quality model for HTTP adaptive streaming: ITU-T P.1203.1. In: Ninth Int'l. conf. on quality of multimedia experience (QoMEX). IEEE. https://doi.org/10.1109/QoMEX.2017.7965631
39. Rainer B, Lederer S, Müller C, Timmerer C (2012) A seamless Web integration of adaptive HTTP streaming. In: 2012 Proceedings of the 20th European signal processing conference (EUSIPCO). IEEE, pp 1519–1523
40. Rizzo L (1997) Dummynet: a simple approach to the evaluation of network protocols. ACM SIGCOMM Comput Commun Rev 27(1):31–41. https://doi.org/10.1145/251007.251012
41. Robitza W, Göring S, Raake A, Lindegren D, Heikkilä G, Gustafsson J, List P, Feiten B, Wüstenhagen U, Garcia MN, Yamagishi K (2018) HTTP adaptive streaming QoE estimation with ITU-T rec. P. 1203: open databases and software. In: Proceedings of the 9th ACM multimedia systems conf., pp 466–471. https://doi.org/10.1145/3204949.3208124
42. Rusan A, Vasiu R (2015) Emulation of backhaul packet loss on the LTE S1-U interface and impact on end user throughput. In: 2015 IEEE Int'l. conf. on intelligent computer communication and processing (ICCP). IEEE, pp 529–536. https://doi.org/10.1109/ICCP.2015.7312715
43. Ryu E-S, Ryu S (2017) Robust real-time UHD video streaming system using scalable high efficiency video coding. Multimed Tools Applic 76(23):25511–25527. https://doi.org/10.1007/s11042-017-4835-2
44. Scharf M, Kiesel S (2006) NXG03-5: head-of-line blocking in TCP and SCTP: analysis and measurements. In: IEEE Globecom 2006. IEEE, pp 1–5. https://doi.org/10.1109/GLOCOM.2006.333
45. Sieber C, Hoßfeld T, Zinner T, Tran-Gia P, Timmerer C (2013) Implementation and user-centric comparison of a novel adaptation logic for DASH with SVC. In: 2013 IFIP/IEEE International symposium on integrated network management (IM 2013). IEEE, pp 1318–1323
46. Spiteri K, Urgaonkar R, Sitaraman RK (2016) BOLA: near-optimal bitrate adaptation for online videos. In: IEEE INFOCOM 2016 - the 35th annual IEEE int'l. conf. on computer communications. IEEE, pp 1–9. https://doi.org/10.1109/INFOCOM.2016.7524428

47. Thang TC, Le HT, Pham AT, Ro YM (2014) An evaluation of bitrate adaptation methods for HTTP live streaming. IEEE J Sel Areas Commun 32(4):693–705. https://doi.org/10.1109/JSAC.2014.140403
48. Van der Hooft J, Petrangeli S, Wauters T, Huysegems R, Alface PR, Bostoen T, De Turck F (2016) HTTP/2-based adaptive streaming of HEVC video over 4G/LTE networks. IEEE Commun Lett 20(11):2177–2180. https://doi.org/10.1109/LCOMM.2016.2601087
49. Wang C, Bhat D, Rizk A, Zink M (2017) Design and analysis of QoE-aware quality adaptation for DASH: a spectrum-based approach. ACM Trans Multimed Comput Commun Applic (TOMM) 13(3s):45. https://doi.org/10.1145/3092839
50. Wei S, Swaminathan V (2014) Low latency live video streaming over HTTP 2.0. In: Network and operating system support for digital audio and video workshop (NOSSDAV). ACM, p 37. https://doi.org/10.1145/2597176.2578277
51. Wisniewski P, Beben A, Batalla JM, Krawiec P (2015) On delimiting video rebuffering for stream-switching adaptive applications. In: 2015 IEEE Int'l. conf. on communications (ICC). IEEE, pp 6867–6873. https://doi.org/10.1109/ICC.2015.7249420
52. Yadav PK, Shafiei A, Ooi WT (2017) Quetra: a queuing theory approach to dash rate adaptation. In: Proceedings of the 25th ACM international conference on multimedia, pp 1130–1138
53. Yahia MB, Louedec YL, Simon G, Nuaymi L, Corbillon X (2019) HTTP/2-based frame discarding for low-latency adaptive video streaming. ACM Trans Multimed Comput Commun Applic (TOMM) 15(1):18. https://doi.org/10.1145/3280854
54. Ye Y, Andrivon P (2014) The scalable extensions of HEVC for ultra-high-definition video delivery. IEEE MultiMedia 21(3):58–64. https://doi.org/10.1109/MMUL.2014.47
55. Zabrovskiy A, Feldmann C, Timmerer C (2018) Multi-codec DASH dataset. In: Proceedings of the 9th ACM multimedia systems conf., pp 438–443. https://doi.org/10.1145/3204949.3208140
56. Zhang T, Chowdhery A, Bahl P, Jamieson K, Banerjee S (2015) The design and implementation of a wireless video surveillance system. In: Proceedings of the 21st annual international conference on mobile computing and networking, pp 426–438. https://doi.org/10.1145/2789168.2790123
57. Zhou C, Lin C-W, Zhang X, Guo Z (2013) A control-theoretic approach to rate adaption for DASH over multiple content distribution servers. IEEE Trans Circuits Syst Video Technol 24(4):681–694. https://doi.org/10.1109/TCSVT.2013.2290580