Received 12 September 2022, accepted 9 October 2022, date of publication 14 October 2022, date of current version 20 October 2022.

Digital Object Identifier 10.1109/ACCESS.2022.3214827

RESEARCH ARTICLE

DoFP+: An HTTP/3-Based Adaptive Bitrate Approach Using Retransmission Techniques

MINH NGUYEN, (Member, IEEE), DANIELE LORENZI, (Member, IEEE), FARZAD TASHTARIAN, (Member, IEEE), HERMANN HELLWAGNER, (Senior Member, IEEE), AND CHRISTIAN TIMMERER, (Senior Member, IEEE)

Christian Doppler Laboratory ATHENA, University of Klagenfurt, 9020 Klagenfurt am Wörthersee, Austria

Corresponding author: Minh Nguyen (minh.nguyen@aau.at)

This work was supported in part by the Austrian Federal Ministry for Digital and Economic Affairs; in part by the National Foundation for Research, Technology and Development; in part by the Christian Doppler Research Association; and in part by the Christian Doppler Laboratory ATHENA (https://athena.itec.aau.at/).

ABSTRACT *HTTP Adaptive Streaming* (HAS) solutions use various adaptive bitrate (ABR) algorithms to select suitable video qualities with the objective of coping with the variations of network connections. HTTP has been evolving with various versions and provides more and more features. Most of the existing ABR algorithms do not significantly benefit from the HTTP development when they are merely supported by the most recent HTTP version. An open research question is "*How can new features of the recent HTTP versions be used to enhance the performance of HAS*?" To address this question, in this paper, we introduce **Days of Future Past+** (DoFP+ for short), a heuristic algorithm that takes advantage of the features of the latest HTTP version, HTTP/3, to provide high Quality of Experience (QoE) to the viewers. DoFP+ leverages HTTP/3 features, including *(i) stream multiplexing*, *(ii) stream priority*, and *(iii) request cancellation* to upgrade low-quality segments in the player buffer while downloading the next segment. The qualities of those segments are selected based on an objective function and throughput constraints. The objective function takes into account two factors, namely the *(i) average bitrate* and the *(ii) video instability* of the considered set of segments. We also examine different strategies of download order for those segments to optimize the QoE in limited resources scenarios. The experimental results show an improvement in QoE by up to 33% while the number of stalls and stall duration for DoFP+ are reduced by 86% and 92%, respectively, compared to state-of-the-art ABR schemes. In addition, DoFP+ saves, on average, up to 16% downloaded data across all test videos. Also, we find that downloading segments sequentially brings more benefits for retransmissions than concurrent downloads; and lower-quality segments should be upgraded before other segments to gain more QoE improvement. Our source code has been published for reproducibility at https://github.com/cd-athena/DoFP-Plus.

INDEX TERMS HTTP/3, ABR algorithm, QoE, HAS, DASH.

I. INTRODUCTION

Video streaming has been growing tremendously in global Internet traffic. Video data occupancy in 2017 accounted for 77% of global Internet traffic and is expected to reach 82% by the end of 2022 [1]. *HTTP Adaptive Streaming* (HAS) is one of the predominant techniques to deliver video data [2], [3]. In HAS, the video at the server side is encoded at multiple quality levels, each of which is represented by a specific bitrate, then split temporally into segments with the same duration. At the client side, an adaptive bitrate (ABR) algorithm is in charge of selecting and requesting the suitable bitrate of each segment to adapt to fluctuations of the network throughput while aiming at providing the best possible Quality of Experience (QoE) [4]. The ABR algorithms can be grouped as: *(i)* throughput based, *(ii)* buffer based, and *(iii)* hybrid adaptation [3]. The existing approaches often make a decision on the quality level for the next segments

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

according to the user's context conditions (*e.g.*, buffer occupancy and network conditions). Low-bitrate segments can be downloaded at times of low throughput to avoid stall events. Around 36% of sessions reported in [5] experience this issue. Low-quality content played out understandably annoys the viewer. This issue may be solved by replacing those segments with higher-quality versions (*i.e.*, performing segment retransmissions) when the network throughput becomes more favorable. Furthermore, to optimize this retransmission process, the segment delivery can be enhanced by relying on several features of the newest versions of HTTP [5], [6], [7].

The third version of HTTP (HTTP/3) is in the process of standardization by IETF [8]. HTTP/3 built on QUIC [9] inherits the *stream multiplexing* feature, with which multiple independent streams can be processed simultaneously over a single connection. In addition, this HTTP version retains key features from its predecessor, HTTP/2 [10]: *(i) server push*, *(ii) stream priority*, and *(iii) request cancellation*. *Server push* enables the client to download multiple segments by sending a single request. This feature is validated in HTTP/2 to reduce request overhead and latency [11], [12]. The client uses *stream priority* to express which data is more preferred and should be downloaded faster than the others when multiple data streams are open concurrently. In case the data conveyed by a stream will not be used after its arrival, the client is able to terminate that stream with *request cancellation* so that the throughput will not be wasted for delivering those unused data. Moreover, HTTP/2 uses the Transmission Control Protocol (TCP) as the transport protocol, so it suffers from the Head-of-Line (HoL) blocking issue when packets are lost. Meanwhile, QUIC runs on top of the User Datagram Protocol (UDP). It mitigates the HoL blocking problem, and HTTP/3 benefits from this [13], [14].

QUIC has been deployed globally in the servers of Google and Akamai [15]. And HTTP/3 has been enabled for more and more websites. By the end of 2020, less than 5% of the websites used HTTP/3; but currently, this figure has increased substantially and reached 25%.[1] However, those websites often host web page objects from third-party HTTP/2 servers [16].

Though HTTP/3 has attracted attention from researchers in the literature, it has been primarily evaluated in the default mode where none of the above features are used [16], [17], [18], [19]. The question of "*What can one do with HTTP/3 to enhance HAS performance?*" has not been fully answered [20].

This paper introduces an HTTP/3-based ABR approach, namely **Days of Future Past+**[2] (DoFP+), to cope with the aforementioned problems. DoFP+ takes into account the qualities of the next segment and the so-called retransmitted/upgraded segments (higher-quality versions of low-quality segments currently located in the buffer), then sends requests to download them using HTTP/3's features. These qualities are selected based on an objective function that maximizes the average bitrate while minimizing the video instability and stall risk. DoFP+ returns a list of bitrates and the corresponding segment indexes of the video that need to be (re)downloaded. It leverages *stream multiplexing* to send multiple requests at the same time and utilizes *stream priority* to download higher-prioritized segments before the others. The priority order of the segments is determined based on our findings of the impact of segment qualities on the users' QoE. While redownloading upgraded segments, if the throughput is unfavorable so that those segments are expected not to be received by the client in time, or the buffer occupancy is going down, DoFP+ will terminate those segments using the *request cancellation* feature.

In general, the advantages of the DoFP+ method are three-fold:

- Existing conventional ABR algorithms do not use these new features and only rely on HTTP/1.1. Meanwhile, DoPF+ takes advantage of key features of HTTP/3 including *stream multiplexing*, *stream priority*, and *request cancellation* to improve the QoE.
- Existing conventional ABR algorithms simply "ignore" improving the low-quality segments downloaded to the buffer. These segments degrade the QoE of the user. Meanwhile, DoFP+ tries to enhance the quality of these segments so that the user is not annoyed by watching low-quality patterns of the video.
- We investigated different strategies for downloading upgraded segments with HTTP/3 features. DoFP+ takes advantage of our findings and utilizes the most effective way to upgrade low-quality segments to achieve the best QoE.

The contributions of this paper are four-fold:

- **Retransmission order analysis:** We evaluate different strategies of download order for the upgraded segments and select the one that optimizes the success of downloading and reduces the rebuffering risk.
- **Utilization of the extensible prioritization scheme for HTTP:** We provide a use-case where the extensible prioritization scheme in [22] can be used efficiently in HAS.
- **DoFP+:** We propose an HTTP/3-based ABR algorithm that takes advantage of key features of HTTP/3 including *stream multiplexing*, *stream priority*, and *request cancellation*. DoFP+ is able to improve the performance of HAS by optimizing the average quality and minimizing video instability and stall duration.
- **Evaluation:** Various experiments are conducted to validate our proposed method while streaming different test videos and provide substantial findings regarding segment upgrades.

The paper is organized as follows. Section II overviews related work in the context of HAS, followed by problem description and motivation of our work in Section III. The proposed method, DoFP+, is described in Section IV.

[1]https://w3techs.com/technologies/details/ce-http3. Accessed 1 March 2022.

[2]Days of Future Past is the title of a superhero film [21] about a story of traveling to the past and changing some events to have a better future.

In Section V, we compare the performance of DoFP+ with state-of-the-art ABR algorithms. Finally, Section VI concludes the paper.

II. RELATED WORK

The literature provides various ABR algorithms. Based on the metrics used for adapting the bitrate selection, the ABR methods can be classified into three groups: *(i)* the throughput-based ABR algorithms rely on the estimation of the network capacity to decide which version of the next segment needs to be requested; *(ii)* the buffer-based ABR algorithms count instead on the state of the player's buffer and predictions of the future buffer occupancy to choose the quality of the next segment to be fetched; lastly *(iii)* the hybrid ABR approaches pick the quality of the next segment based on a combination of instant or smoothed metrics, *e.g.*, throughput and buffer occupancy, or predictions of future states, *e.g.*, estimation of the buffer situation after the download of a specific segment [3].

In this paper, we compare different ABR approaches from these classes with DoFP+.

Among the throughput-based ABR algorithms, we examine the approach proposed by Nguyen et al. [23]. It aggressively (hence, the acronym AGG) selects the quality level whose bitrate is the maximum and smaller than the estimated throughput.

In the class of buffer-based approaches, BBA-0 [24] and BOLA [25] rely on instant buffer occupancy to select which quality version of the next segment needs to be requested. BBA-0 [24] requests the version of the next segment whose bitrate matches a monotonic non-increasing function of the instant buffer state. Furthermore, two thresholds must be provided, namely a low and a high threshold. When the buffer occupancy is lower than the first threshold, BBA-0 requests the lowest-quality version. If otherwise, the buffer level is higher than the latter, the client requests the highest-quality version. In this manner, the ABR algorithm relies on a piece-wise linear function. Focusing on the buffer occupancy and omitting a complex throughput estimation task, BBA-0 requires a broad buffer to cope with throughput fluctuations. BOLA [25] is an online control algorithm using Lyapunov optimization. It requests the quality version of the next segment by solving a utility maximization problem. Specifically, BOLA selects the quality level which jointly maximizes the playback utility, related to segments' qualities and playback smoothness, *i.e.*, streaming time that is spent not rebuffering. A high-quality segment provides higher playback utility, but the download of this segment is likely to impact in a negative way the playback smoothness leading to stalls in video reproduction.

Regarding hybrid schemes, Juluri et al. [26] propose SARA, an ABR algorithm that retrieves the next segment based on both current buffer state and throughput estimation. SARA's strategy depends on the subdivision of the buffer capacity into three areas from low to high, identified by the parameters I, B_α, B_β ($I < B_\alpha < B_\beta$). When the buffer level occurs to be smaller than I, the "fast start" phase induces the selection of the lowest quality version. Otherwise, if the buffer is not greater than B_α, the quality level of the next segment will be increased by one unit, following the so-called "additive increase" method. The optimal working area, according to SARA, is when the buffer state value is between B_α and B_β. In this area, the quality level is increased or stabilized based on the network conditions and the buffer level. Otherwise, the best suitable quality for the throughput is chosen, and the download is delayed with the goal of forcing the buffer to operate in the optimal working area.

HTTP/3 and QUIC have been recently studied in the context of *HAS*. Timmerer et al. [18] investigate Dynamic Adaptive Streaming over HTTP (DASH) performance over QUIC (v19) and state that this protocol cannot bring benefit to the overall streaming performance. Furthermore, Seufert et al. [27] compare QUIC's and TCP's behaviors in the context of video streaming *in the wild*. For this purpose, Google services and servers have been accessed like an end user would. The reported results seem to evidence no QoE-relevant improvement while switching from TCP to QUIC. Bhat et al. [18] reinforce that finding by implementing common ABR approaches over QUIC and TCP. The authors assert that traditional ABRs originally designed for TCP connections do not achieve any improvement in QUIC connections. Arisu et al. [28] test QUIC (v39) in a specific scenario and report some contributions of QUIC. When the network interface changes frequently and users send frame-seek requests, QUIC enhances QoE due to the reduction in the wait time and rebuffering rates. Perna et al. [16] evaluate HTTP/3 performance in the context of adaptive video streaming with Cloudflare's *quiche* HTTP/3 and QUIC implementation.[3] The authors conclude that HTTP/3 does not bring benefits to the QoE of viewers. In these works, however, QUIC and HTTP/3 are evaluated in the default mode only, and no key features are exploited.

Bhat et al. [5] leverage HTTP/3's *stream multiplexing* feature to additionally upgrade low-quality segments in the buffer, besides the next segment, for the purpose of enhancing the QoE. The work in [29] evaluates a retransmission technique called H2BR [7], originally designed for HTTP/2 and non-scalable coded video streaming, in the context of scalable coded video streaming (SVC) over HTTP/3. H2BR utilizes *server push*, *stream multiplexing*, *stream priority*, and *request cancellation* to select suitable qualities for upgraded segments after the ABR makes a decision on the quality for the next segment. A segment is upgraded by an additional enhancement layer in SVC [30] or a higher-bitrate segment in the non-scalable coded video. Both these works are dependent on the ABR algorithm deployed at the client; hence, their performances vary according to the different ABRs.

Our previous work in [6] introduced the Days of Future Past (DoFP) approach that relies on a Mixed Integer Linear Programming (MILP) model to determine the qualities

[3]https://docs.quic.tech/quiche/

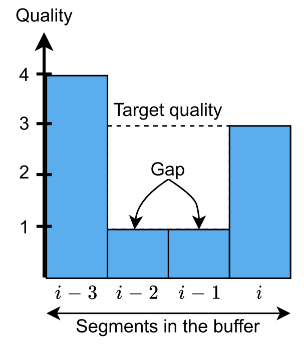


FIGURE 1. Examples of quality gaps.

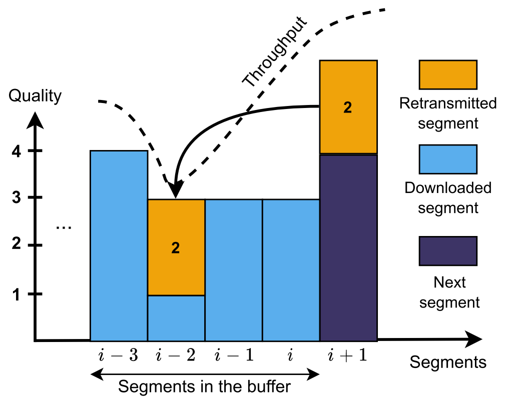


FIGURE 2. Motivation of retransmission.

for the next and upgraded segments. A set of qualities that maximize the minimum quality of those segments and the sum of qualities will be selected. A combination of HTTP/3's *stream multiplexing* and *request cancellation* mechanisms is validated to bring benefits for the QoE. However, *stream priority* is not used in this work, so the concurrent segments share the throughput equally.

In this paper, our proposed DoFP+ is a heuristic approach that finds the qualities for the segments that maximize the average video quality while minimizing video instability and stall risk. All segments in the buffer and the next segment are considered simultaneously, and multiple requests can be sent at the same time. Different from DoFP, our DoFP+ scheme leverages additionally the *stream priority* feature of HTTP/3. And, unlike H2BR, where a complex and limited-deployment design for this feature is used, we utilized a new prioritization scheme proposed in [22]. This scheme is implemented in HTTP/3 and might be integrated into HTTP/2's revision [31]. In addition, instead of downloading upgraded segments concurrently, DoFP+ processes them sequentially based on our profound analysis of various sorts of download order.

III. PROBLEM DESCRIPTION AND MOTIVATION

A. QUALITY GAP DEFINITION AND TARGET QUALITY

We define a segment group as a set of one or multiple adjacent segments in the buffer which have the same quality level. Fig. 1 depicts two plausible versions (left and right) of a set of segments stored in the buffer at a certain time t. Let i denote the index of the last downloaded segment. A segment group is a quality gap if it belongs to one of the following scenarios:

- The quality level of the segments in the group is lower than the quality levels of the two segments adjacent to the group, *i.e.*, the segment before the first one forming the group and the segment after the last one belonging to the group; see Fig. 1 (left).
- The quality level of the segments in the last group is lower than that of the previous group, as shown in Fig. 1 (right).

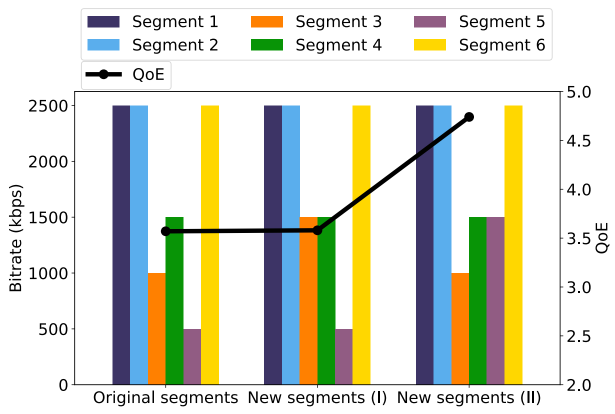
The existence of quality gaps in a set of buffered segments can seriously impact the quality smoothness and perceived QoE of a streaming session. Therefore, if the available throughput is higher than the requested bitrate for the next segment, the client should try filling at least one (if many are present) of the quality gaps. To fill a specific gap while simultaneously avoiding strong interference with the download of the next segment, the target quality level chosen for the retransmission of the low-quality segments in each gap should not be too high. Thus, DoFP+ upgrades these segments by selecting a quality level that is not higher than the lower quality level of their adjacent group, as illustrated in Fig. 1.

B. MOTIVATION

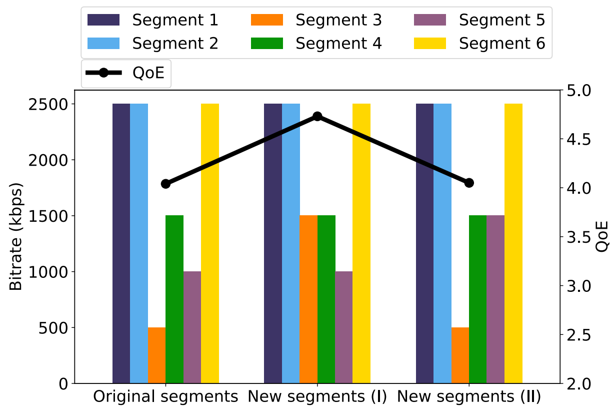
Fig. 2 shows an example of the buffered segments with quality variations due to throughput fluctuation. Four segments from segment $i-3$ to segment i in the buffer have different quality levels 4, 1, 3, and 3, respectively. Thus, there are three groups of segments corresponding to quality levels 4, 1, and 3. Clearly, the second group with segment $i-2$ in quality 1 is a quality gap. A traditional ABR only selects the quality for the next segment (*i.e.*, segment $i+1$) with the highest level (*i.e.*, quality 4) as the best decision, but the viewer still experiences the quality gap no matter how high the throughput is. This results in a low QoE because of the quality gap [32].

In addition, in a streaming session, some ABR algorithms generally force the client to stop downloading the next segments to prevent a full buffer, which is called OFF periods [33]. This strategy leads the client to lose information about the throughput. Thus, a wrong decision can be made if the throughput is overestimated.

Retransmission is supposed to tackle these issues. The retransmitted segments can be downloaded even when the buffer is full as they replace the ones in the buffer and do not increase the buffer occupancy. Consequently, the number of OFF periods is decreased. Moreover, retransmission can upgrade low-quality segments in the buffer to enhance the QoE. For example, in Fig. 2 when the throughput is favorable, not only does the client download the next segment with the



(a) The quality of the first gap is higher than that of the second gap.



(b) The quality of the second gap is higher than that of the first gap.

FIGURE 3. QoE scores of different scenarios to fill quality gaps.

highest quality level, but also it redownloads segment 2 with quality 3 to fill the quality gap.

Consider multiple upgraded segments that are requested at the same time as the next segment. In this case, the question arises in which order these segments will be delivered. In the following section, we will analyze various strategies for download order and select the one that brings the most benefits.

IV. DoFP+ APPROACH

A. RETRANSMISSION ORDER ANALYSIS

We evaluate different possible orders of segment retransmissions. Then, we decide the one that provides the optimal result.

Let a set of quality gaps, each consisting of one or multiple segments, occur throughout a streaming session. When we upgrade the segments in those quality gaps, two questions arise: *(i)* Which quality gap should be filled? *(ii)* Which transmission order for the segments belonging to the selected quality gap should be considered? It should be noted that we fill only one quality gap at a time to decrease request overhead and to make the process less complicated.

TABLE 1. Priority parameters of segments in download strategies.

Download strategies	Stream multiplexing?	Segment priority (u, i)			
		A	B	C	N
ABCN(N)	✗	-	-	-	-
ABCN(M)	✓	(1,0)	(2,0)	(3,0)	(4,0)
CBAN	✓	(3,0)	(2,0)	(1,0)	(4,0)
NABC	✓	(2,0)	(3,0)	(4,0)	(1,0)
NCBA	✓	(4,0)	(3,0)	(2,0)	(1,0)
CONCURRENCE	✓	(1,1)	(1,1)	(1,1)	(1,1)

1) QUALITY GAP ORDER

Consider several quality gaps in the buffer. We evaluate different strategies for upgrading the gaps and find the one that gets the optimal QoE. The QoE scores are calculated by the ITU-T P.1203 model[4] [34], [35] with segment duration 4 seconds, the H.264/AVC video codec and 1080p screen size. The pairs of bitrate and resolution are (500kbps, 360p), (1000kbps, 720p), (1500kbps, 720p), and (2500kbps, 1080p).

Assume that there are six segments located in the buffer with two quality gaps: one composed of segment index 3 and one containing segment index 5, which are shown as *Original segments* in Fig. 3 and labeled consistently. In Fig. 3a, segment index 3, *i.e.*, the first quality gap, has higher quality/bitrate than segment index 5, *i.e.*, the second quality gap, (1000kbps vs. 500kbps). In contrast, the bitrate of the first quality gap in Fig. 3b is less than that of the second quality gap. The strategy *New segments (I)* fills the first quality gap, whereas *New segments (II)* upgrades the second quality gap with the same target quality. It can be seen that the QoE achieves more gain when the quality gap with lower quality is filled. Especially, upgrading the segment from 1000kbps to 1500kbps hardly improves the QoE score. For example, in Fig. 3b the QoE scores of *Original segments* and *New segments (II)* differ by less than 0.3%, although the latter strategy downloads additionally a 1500kbps segment. On the other hand, with the same additional segment quality/bitrate but for substituting the lower quality gap (*i.e.*, from 500kbps to 1500kbps) by the *New segments (I)* strategy, the QoE score is increased by 17% from 4.04.

Therefore, we conclude that the quality gap with the lower quality level should get higher priority.

2) SEGMENT ORDER

As discussed in Section III-B, the proposed approach, under certain conditions, grants the client the opportunity to request higher quality versions of already-stored segments in the buffer, with the purpose of improving the QoE-related metrics. The retransmission strategy as such, *i.e.*, the selection of segments and respective qualities involved in the retransmission process, is part of the proposed algorithm and is covered in Section IV-B. The question we plan to investigate here is *which transmission order for the next segment and the segments belonging to the selected quality gap should be considered*. Indeed once the selection is completed, and

[4]https://github.com/itu-p1203/itu-p1203

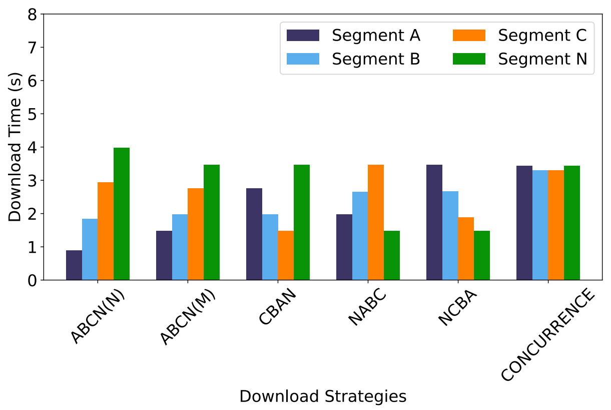


FIGURE 4. Download time of different download strategies with 100ms RTT and 0% packet loss rate.

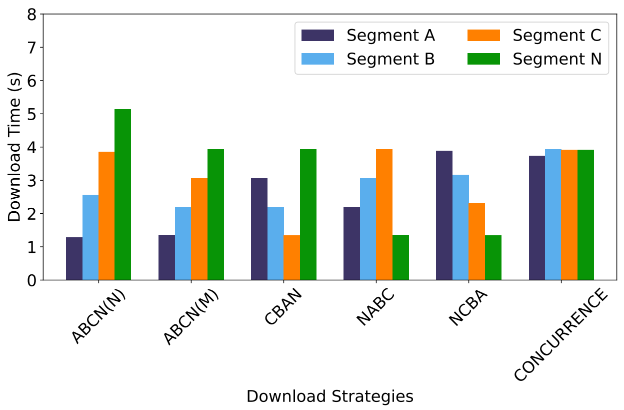


FIGURE 6. Download time of different download strategies with 400ms RTT and 0% packet loss rate.

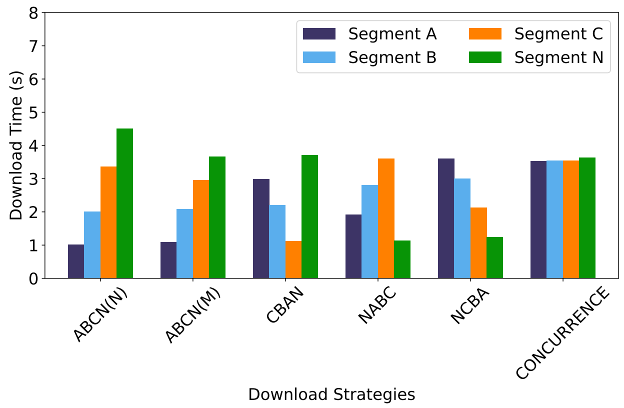


FIGURE 5. Download time of different download strategies with 100ms RTT and 5% packet loss rate.

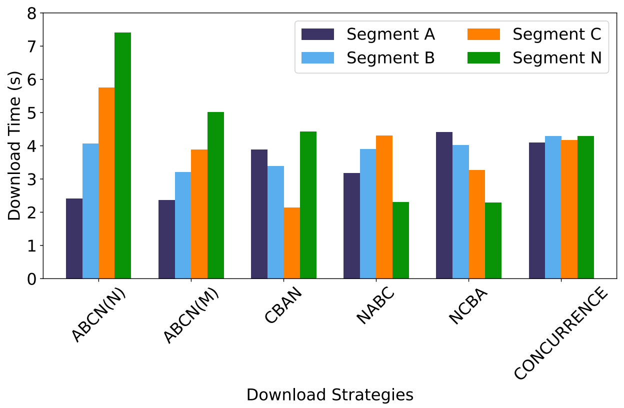


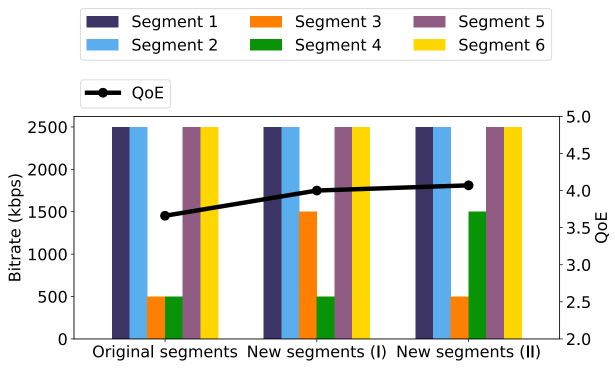
FIGURE 7. Download time of different download strategies with 400ms RTT and 5% packet loss rate.

the client is on the verge of requesting the chosen segments, the system must determine a priority queue to start the transmission. In HTTP/3 the priority mechanism relies on the two parameters called *urgency* and *incremental* to be modified in the HTTP request headers [22]. The *urgency* (u) of a requested resource defines its importance in the transmission queue and is represented by an integer number from 0 to 7, where lower values imply higher importance. The *incremental* (i) parameter is described by a Boolean value indicating if an HTTP response can be processed incrementally, *i.e.*, provide some meaningful output as chunks of the response arrive. In this case, the overall available bandwidth is usually distributed (equally) between incremental responses that share the same urgency.

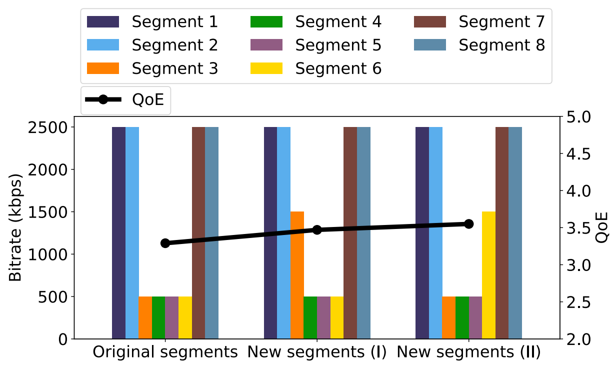
In order to identify the desired combination of these two parameters in our use-case environment, we developed a testbed using an HTTP/3-QUIC implementation, LSQUIC,[5] to analyze different transmission order patterns. In our experiments, we considered segments with 4s length and assumed the client buffer to contain three consecutive segments, namely A, B, and C, to be retransmitted with a bitrate of 1Mbps each; the next segment N must be fetched at the same 1Mbps bitrate. The mentioned testbed includes the following download strategies as listed in Table 1 (ABCN means that A will be requested first and then B, C and N will follow): ABCN(N), without the *stream multiplexing* feature; ABCN(M), CBAN, NABC and NCBA with the *stream multiplexing* feature; and eventually concurrent transmissions (*i.e.*, CONCURRENCE.) of the segments (same *urgency* value for all requests and *incremental* parameter set to true). The results are then evaluated through different combinations of network parameters by means of download time for each segment. The throughput is set to 5Mbps, the RTT parameter changes from 100 to 400 ms, and the packet loss rate can be set to 0 or 5% of the total transmitted packets.

Fig. 4, 5, 6 and 7 represent the download time of segments A, B, C and N based on the selected strategy in different network scenarios. Specifically, Fig. 4 and 5 show the results for a fixed 100ms RTT and a packet loss rate of 0% and 5%, respectively, whereas Fig. 6 and 7 illustrate the trend for a fixed 400ms RTT and a packet loss rate equal to 0% and 5%, respectively. Analyzing these figures, we notice how the *no multiplexing* strategy suffers from high RTT values and packet loss rates more than the other strategies. In the

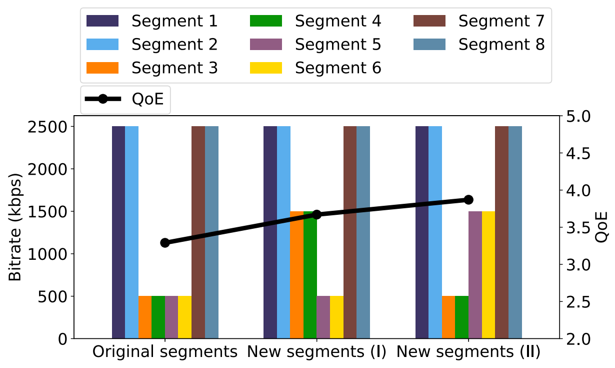
[5]https://github.com/litespeedtech/lsquic.



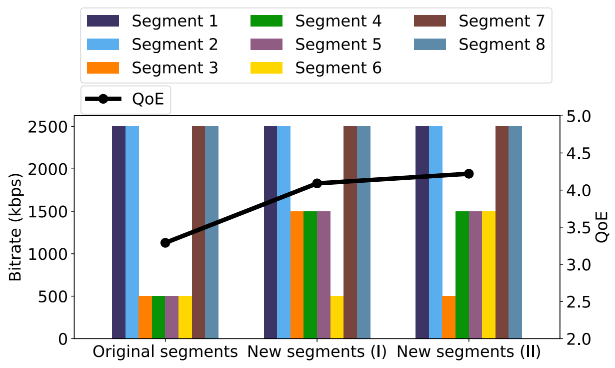
(a) Upgrade one segment in a two-segment quality gap.



(b) Upgrade one segment in a four-segment quality gap.



(c) Upgrade two segments in a four-segment quality gap.



(d) Upgrade three segments in a four-segment quality gap.

FIGURE 8. QoE scores of different download orders of segments in a quality gap.

0% packet loss scenario depicted in Fig. 4, ABCN(N) ends the transmission in 3.9s whereas ABCN(M) with *stream multiplexing* requires only 3.5s. With the increase of packet loss to 5% (Fig. 5), compared to the no packet loss environment ABCN(N) takes 4.5s (+13%) to complete the downloads while ABCN(M) terminates the transmission in 3.7s (+5.6%). In this case, the reduction in the download time for ABCN(M) compared to ABCN(N) counts for ∼19%. This margin between the two strategies is due to how the requests are sent and managed. Each multiplexed segment corresponds in HTTP/3 to a separate independent stream, which helps to mitigate the HoL blocking issue - from which the previous HTTP versions heavily suffer - thanks to the QUIC protocol as the transport layer. This explains the major resistance of the download strategies with *stream multiplexing* towards the increment of packet loss rate.

The difference between ABCN(N) and ABCN(M) is even more visible, switching RTT from 100ms to 400ms, not considering the packet loss rate. Indeed, comparing Fig. 4 and 6, one sees a 29% growth for ABCN(N) (5.1s) and a 14% increase for ABCN(M) (3.9s). The non-adoption of the *stream multiplexing* feature severely impacts the download time of ABCN(N) with an increase from 15% (Fig. 4) to 31% (Fig. 6) with respect to the same strategy exploiting the HTTP/3 feature. Without *stream multiplexing* the client needs to wait for a segment to be correctly received before sending the next request, which clearly introduces a delay equal to $m \times$RTT - where m is the number of segments to be requested. However, adopting *stream multiplexing* the client sends all the requests at once and waits for the server to deliver all the requested segments, reducing the delay to only one RTT.

Within this use-case testbed, with the segments A, B, C, and N sharing the same bitrate, the theory asserts that there is no difference in the download time for multiplexing strategies and the concurrent one. The results practically confirm the theory. We can state that the download time for the presented scenarios and strategies is contained within a tight interval. Analyzing Fig. 4 we find a lower bound in the concurrent strategy with 3.4s and an upper bound in CBAN with 3.5s; for Fig. 5 the results range between 3.6s (NABC) and 3.7s (CBAN); in Fig. 6 every strategy scores around 3.9s; and eventually Fig. 7 concludes the comparison with 4.3s for the concurrent strategy and 4.4s for ABCN.

Among the multiplexing strategies, the concurrent download mimics the expression "*grasp all, lose all*". Indeed trying to fetch all the segments simultaneously may result in useless redownloads, given that the segments in the buffer have different playback timings, and some segments may arrive later than the relative playout deadlines. However we need the segments to be differently prioritized to *(i)* reduce the risk of rebuffering events, *i.e.*, the next segment must be fetched first, and *(ii)* maximize the QoE provided by the selected segments and download strategy.

Based on the first consideration, both NACB and NCBA strategies are plausible. The selection of the optimal strategy for the proposed method must be considered in the context of our QoE-related findings. As an example, we will show which segment in a quality gap should be upgraded when the client has enough resources (*i.e.*, throughput and buffer occupancy) to additionally download only one segment beside the next segment.

Consider the scenario depicted in Fig. 8 in which several segments with different quality levels are stored in the client's buffer. Fig. 8 introduces two upgrade options, depicted as *New segments (I)* and *New segments (II)*, and furthermore illustrates the resulting QoE scores for the presented sequences of segments. Here, we answer the question "*Should the segments located in a quality gap be upgraded in ascending or descending order of playout time?* to optimize the QoE?". We do not redownload segments in the middle of the quality gap as it creates new quality switches, which clearly impair the QoE [32].

Fig. 8a considers the scenario of two segments if a quality gap. The original segments in the buffer have a QoE score of 3.66, and if a 1500kbps segment is downloaded to upgrade the third segment as in *New segments (I)*, there is an increase in the QoE score by 0.36. On the other hand, using this 1500kbps segment to replace the fourth segment like *New segments (II)* provides the best performance with a 4.07 QoE score. Thus, the later-played segment should be upgraded first.

Fig. 8b, Fig. 8c, and Fig. 8d investigate a quality gap consisting of four segments. If there is enough resource to upgrade one segment (Fig. 8b), the last segment in the quality gap (*i.e.*, segment 6) should be selected rather than the first one (*i.e.*, segment 3) to gain better QoE (3.55 vs. 3.47). Similar results can be captured when more segments are upgraded as shown in Fig. 8c and Fig. 8d. It can be attributed to the number of downward switches. A downward (upward) switch happens when the quality is decreased (increased) between two segments. When an earlier-played segment is upgraded, it results in a downward switch, whereas upgrading a later-played segment generates an upward switch if the quality gap is not fully filled. Because viewers feel less annoyed by an upward switch than a downward one [36], we achieve better QoE improvement when we download later-played segments.

These outcomes can be generalized to an arbitrary number of segments. The maximization of the QoE reflects the strategy presented as NCBA, on which the proposed method will build the actual download strategy.

From the above analysis, we propose a strategy abiding by the following download order:

- First, the next segment is downloaded to fill the buffer and reduce stall possibility.
- Second, the quality gaps with the lowest quality level will be considered.
- Third, if there are some quality gaps that have the same lowest quality level, the one which is played out earlier will be upgraded to increase the chance of more successful retransmissions.
- Finally, when the considered quality gap has more than one segment, the later-played segment should be upgraded first in terms of quality.

TABLE 2. The notations used in DoFP+.

Notations	Description
$\mathcal{R}$	The set of quality levels
N	The number of quality levels
τ	The segment duration
$\mathcal{S}$	The set of segments
s_0	The segment playing on the screen
s_k	The next segment to be regularly appended to the buffer
s_i	The segments fully located in the buffer ($0 < i < k$)
$\mathcal{Q}$	The set of quality levels of segments in $\mathcal{S}$
q_i	The quality level of segment s_i
r_i	The current bitrate of segment s_i
$\mathcal{Q}^c$	The set of quality levels returned by DoFP+
r_i^c	The new bitrate of segment s_i selected by DoFP+
q_i^c	The new quality level of segment s_i selected by DoFP+
$F(\mathcal{Q}')$	The objective function of an arbitrary quality set $\mathcal{Q}'$
t_i^{down}	The download time of segment s_i at bitrate r_i
t_i^{avai}	The available time to download segment s_i
t_i^{play}	The playback time of segment i
t^{curr}	The time of the currently played segment
T^e	The estimated throughput
B^c, B^e	The current and estimated buffer level, respectively
B^l, B^s, B^h	The low, safe, and high buffer thresholds, respectively
B^{cancel}	The buffer threshold to terminate redownloaded segments
t^{cancel}	The time threshold to terminate redownloaded segments

B. DoFP+ ALGORITHM

In this section, we describe our proposed DoFP+ algorithm in detail. Table 2 shows the notations used in DoFP+. Assume that the video is encoded into N quality levels indexed as $\{1, 2, \ldots, N\}$ with segment duration τ. Let $\mathcal{S} = \{s_0, s_1, \ldots, s_k\}$ denote a set of consecutive segments consisting of the segment being played out s_0, the ones in the buffer from s_1 to s_{k-1}, and the next segment to be downloaded regularly to the buffer s_k; and $\mathcal{Q} = \{q_0, q_1, \ldots, q_k\}$ is the set of quality levels of those segments. It should be noted that q_k is initially assigned to 0 as the quality of the next segment has not been determined yet. Therefore, there is always at least one quality gap at segment s_k. The output of DoFP+ is an array of chosen qualities $\mathcal{Q}^c$ that maximizes our objective function under the constraint of downloading time.

1) OBJECTIVE FUNCTION AND CONSTRAINTS

The main target of HAS is to provide viewers with optimal QoE by delivering high and stable video quality with minimum stall events [3], [30]. DoFP+ chooses a set of quality levels $\mathcal{Q}^c$ to maximize an objective function $F(\cdot)$. The objective function of an arbitrary set $\mathcal{Q}' = \{q'_0, q'_1, \ldots, q'_k\}$ of quality levels of segments, $F(\mathcal{Q}')$, takes into account two factors: 1) segment quality, and 2) video instability, and can be introduced as Eq. (1).

$$F(Q') = \alpha \times \sum_{q'_n \in Q'} \frac{q'_n}{N} - (1-\alpha) \times \sum_{q'_n, q'_{n+1} \in Q'} \frac{|q'_n - q'_{n+1}|}{q'_{n+1}}, \tag{1}$$

where $\alpha \in [0, 1]$ is the weight of the segment quality factor. Therefore, the chosen set Q^c can be determined as follows:

$$Q^c = \arg\max_{Q'} F(Q') \tag{2}$$

Regarding video instability, we consider not only the changes in the quality of adjacent segments (*i.e.*, quality switch amplitude) but also the final quality value of those changes. The reason is that a specific quality switch amplitude (*i.e.*, $|q'_n - q'_{n+1}|$) results in less annoyance if the final quality (*i.e.*, q'_{n+1}) is at a higher level [37]. For instance, a decrease from quality level 4 to 2 intuitively leads to a worse quality degradation than when the quality jumps down from 6 to 4. However, if DoFP+ solely relies on Eq. (1), then it always selects the highest quality for all segments which with high probability results in stall events as a result of throughput variations. Thus, we consider two constraints to reduce the stall risk.

First, we limit the download time t_i^{down} of segment s_i at bitrate r_i to be less than the available time t_i^{avai} of that segment.

$$t_i^{down} < t_i^{avai}, \forall i \in [1, k] \tag{3}$$

The download time t_i^{down} can be calculated as follows:

$$t_i^{down} = \frac{r_j \times \tau}{T^e}, \tag{4}$$

where T^e is the estimated throughput. T^e can be calculated by a "smooth" throughput [38] and the last measured throughput as in [39]. The next segment s_k contributes a period of segment duration τ to the buffer but also drains the buffer by its download time t_i^{down}. When the buffer level is small, it is necessary to download the next segment within a short time, less than a segment duration, to prevent rebuffering. Therefore, in this case we assign segment duration τ to the available time t_k^{avai} of the next segment s_k. However, in case the buffer is high, we increase the upper bound t_k^{avai} so that DoFP+ can select better qualities while sacrificing a reasonable amount of buffer. The details of calculating t_k^{avai} will be provided in the next section. The available time t_i^{avai} ($i \in [1, k-1]$) of the buffered segments is determined as the difference between the current time and the time those segments are played out on the screen:

$$t_i^{avai} = t_i^{play} - t^{curr}, \forall i \in [1, k-1], \tag{5}$$

where t_i^{play} and t^{curr} denote the playback time of segment s_i and the time of the currently played segment, respectively.

Second, after downloading the next and upgraded segments, the buffer is increased by segment duration τ but drains by the amount of time required for the download of those segments. Let B^e denote the estimated buffer after all the next and upgraded segments are downloaded. It can be calculated as

$$B^e = B^c + \tau - \frac{\sum_{i \in \{j | q_j \neq q_j^c\}} r_i^c \times \tau}{T^e}, \tag{6}$$

where B^c is the current buffer occupancy and $\sum_{i \in \{j | q_j \neq q_j^c\}} r_i^c$ is the total bitrate of the next and upgraded segments. We set B^e to be higher than a threshold that is half of the buffer size (see Section IV-B2) to reduce the stall possibility and rapidly upgrade other segments in the buffer.

Retransmission techniques can waste some network resources as low-quality segments are downloaded but not used by the client. This work, however, focuses on optimizing the QoE of the client. We do not add the network waste in the constraints as it can reduce the improvement of DoFP+.

The details of DoFP+ are provided in the next section.

2) DoFP+ ALGORITHM

We divide the buffer with three thresholds: B^l, B^s and B^h ($B^l < B^s < B^h$). It is worth noting that only B^l and B^s are considered in the selection of the algorithm stage. There are three stages in DoFP+ as follows:

- **Startup stage** ($B^c < B^l$): When the buffer is small, it is substantial to ramp up the buffer to avoid any delay in playing video. Therefore, DoFP+ selects the lowest bitrate.
- **Single-download stage** ($B^l < B^c < B^s$): If the buffer is in a low zone, DoFP+ only focuses on the next segment and selects its quality level that maximizes the objective function in Eq. (1). Upgrading buffered segments is not considered as it would drain the buffer, which frequently makes the buffer jump back to the **Startup** stage.
- **Multiple-download stage** ($B^s < B^c$): In this safe zone of buffer, the client is allowed to redownload (some) segments to enhance their qualities if needed. DoFP+ searches for quality gaps in the segments currently located in the buffer. If there is no quality gap, then only the next segment is downloaded, like in the **Single-download** stage. Otherwise, DoFP+ will additionally select a suitable quality level for other low-quality segments of a quality gap for the purpose of maximizing the objective function.

The proposed ABR algorithm, DoFP+, is presented in Algorithm 1. When $B^c \geq B^s$ (lines 13 – 28), the available download time of the next segment can be assigned with either the segment duration or the difference between current buffer occupancy and the safe threshold (*i.e.*, $B^c - B^s$). When the buffer level is high (*i.e.*, $B^c > B^h$) and only the next segment is downloaded, it is feasible to download the next segment with high quality; hence the available download time t_k^{avai} can be as much as $B^c - B^s$ so that while receiving this segment, the buffer stays above the safe level B^s (lines 15 – 16). Otherwise, we set t_k^{avai} equal to the segment duration to spend time on upgrading low-quality segments in the buffer (lines 17 – 18). This strategy assures that the bitrate for the next segment does not exceed the throughput

Algorithm 1: DoFP+ ABR Algorithm

```
1  Input: $\mathcal{Q}, B^c, T^e, \tau$
2  Output: $\mathcal{Q}^c$
3  for $i$ = 1,2,..., k-1 do
4      $t_i^{avai} = t_i^{play} - t^{curr}$;
5  if $B^c < B^l$ then
6      $\mathcal{Q}^c = \{q_0, q_1, \ldots, q_{k-1}, 1\}$;
7      return $\mathcal{Q}^c$;
8  if $B^c < B^s$ then
9      $t_k^{avai} = \tau$;
10     $q_k^* = \underset{q_k, \; t_k^{down} < t_k^{avai}}{\arg\max} F(\{q_0, q_1, \ldots, q_k\})$;
11     $\mathcal{Q}^c = \{q_0, q_1, \ldots, q_{k-1}, q_k^*\}$;
12     return $\mathcal{Q}^c$;
13 else
14     if no quality gap in the buffer then
15         if $B^c > B^h$ then
16             $t_k^{avai} = B^c - B^s$;
17         else
18             $t_k^{avai} = \tau$;
19         $q_k^* = \underset{q_k, \; t_k^{down} < t_k^{avai}}{\arg\max} F(\{q_0, q_1, \ldots, q_k\})$;
20         $\mathcal{Q}^c = \{q_0, q_1, \ldots, q_{k-1}, q_k^*\}$;
21     else
22         for $q_k' = 1, 2, \ldots, N$ do
23             if $t_k^{download} > t_k^{avai}$ then
24                 break;
25             for each quality gap n in the buffer do
26                 $\mathcal{Q}^n = \underset{t_i^{down} < t_i^{avai}, \; i \in \{j | s_j \in gap\}, \; B^e > B^s}{\arg\max} F(\{q_0, q_1, \ldots, q_k'\})$;
27         $\mathcal{Q}^c = \underset{\mathcal{Q}^n}{\arg\max} F(\mathcal{Q}^n)$;
28 return $\mathcal{Q}^c$;
```

so that we can spend part of the throughput to additionally redownload buffered segments with higher qualities.

After DoFP+ selects the quality for the next and upgraded segments, the client sends multiple requests, each of which corresponds to a segment, at the same time to the server using *stream multiplexing*. The *stream priority* feature will be used by setting the parameters `u` and `i`. The value of `i` is set to 0 or `FALSE` to make the segments come sequentially. The parameter `u` of a request is set based on the order of a segment. From the conclusion about the segment order, we set the latter segment in a quality gap a smaller `u` (*i.e.*, higher priority).

Consider a scenario in which the throughput suddenly drops while the upgraded segments are being delivered. This can result in rebuffering if the client keeps receiving those segments as they do not contribute to the buffer occupancy. To alleviate this issue, DoFP+ executes *request cancellation* of HTTP/3 to terminate the streams delivering the upgraded segments when the current buffer level is low enough or the upgraded segments have a too short time before being played on the screen. For more details, the *request cancellation* feature is invoked when the buffer level B^c should be smaller than a pre-defined threshold, namely B^{cancel}, or the remaining time to play the upgraded segment is less than a certain value, so-called t^{cancel}. Both B^{cancel} and t^{cancel} should be small to avoid frequent cancellation, but B^{cancel} needs to be high enough to ensure a stall will not happen. The values of B^{cancel} and t^{cancel} in our experiments are provided in Section V.

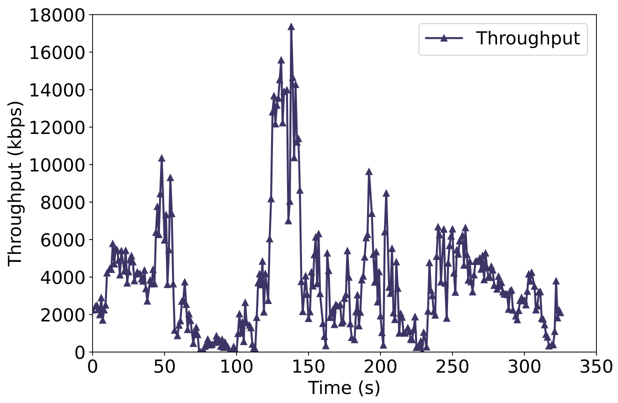


FIGURE 9. Network trace used in our experiments.

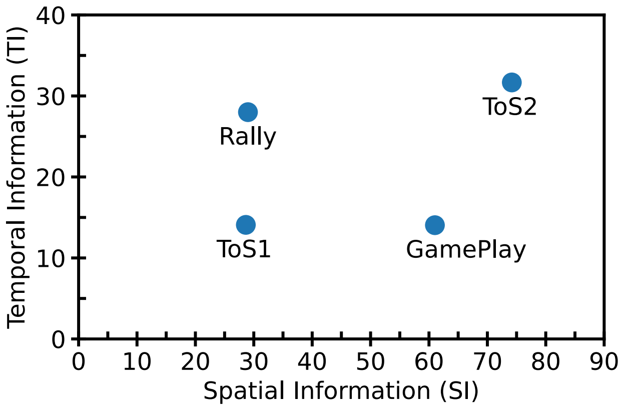


FIGURE 10. SI - TI of the test videos.

V. EXPERIMENTS AND DISCUSSION

A. EXPERIMENTAL SETUP

Our testbed comprises an HTTP/3 server and an HTTP/3 client, which communicate with each other via an Internet connection. Both server and client use the LSQUIC library for HTTP/3 connection on Ubuntu 18.04 LTS. To compare the performance of different ABR approaches, we shape the network connection between the server and the client by the

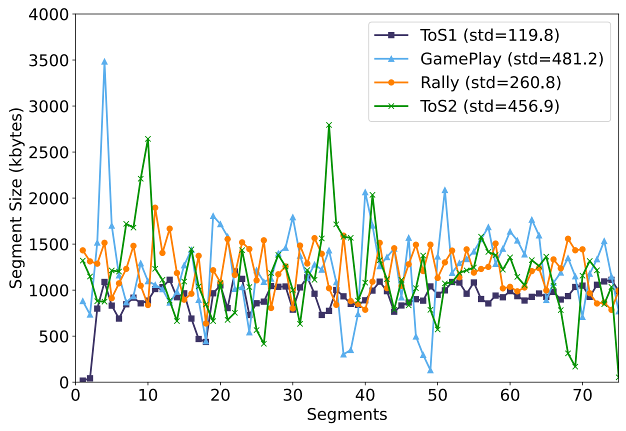


FIGURE 11. **Segment size of test videos at bitrate 2426kbps.**

traffic control (*tc*) tool, according to a network trace collected in [40] as shown in Fig. 9 for all experiments.

We used four test videos stored on the server: *(i)* Tears of Steel (the first 5 minutes) – ToS1, *(ii)* Gameplay[6] [41], *(iii)* Rally,[7] *(iv)* Tears of Steel (the last 5 minutes) – ToS2. These videos have different complexity in terms of spatial information (SI) and temporal information (TI), as illustrated in Fig. 10. As a result, the segment size varies as shown in Fig. 11. It can be seen that segments in the ToS1 video have the most stable size with a standard deviation (std) of 119.8kbytes, whereas this figure for Gameplay is 481.2kbytes. They are encoded into the bitrate ladder $\mathcal{L} = \{107, 240, 346, 715, 1347, 2426, 4121\}$ kbps with corresponding resolutions {144, 240, 360, 480, 720, 1080, 1440}p, respectively [42]. The segment duration τ is 4s as recommended in [43]. At the client, the buffer size B_{max} is set to 20s. The other parameters in the proposed method are set as follows: $\alpha = 0.8$, $t^{cancel} = 100ms$, $B^{cancel} = B^{s} = 0.5 \times B_{max}$. We compare our proposed method DoFP+ with state-of-the-art approaches described in Section II: AGG, BOLA, SARA, BBA-0 (with and without the support of H2BR), and DoFP. Each experiment is run 20 times, and the experimental results represent the average values.

In this paper, we consider the following metrics: *(i) average bitrate* as the average bitrate of all segments played on the screen; *(ii) video instability* denoting the difference in the quality levels of two adjacent segments, calculated as Eq. (7), where n represents the segment index and q_n the quality of the n^{th} segment; *(iii) stall duration* and *number of stalls* expressed as the total period when the video is frozen and the number of times that event occurs, respectively; *(iv) redownloaded data* including the amount of successful data and unsuccessful data that arrives after playout time; *(v) redownloaded segments* specifying the number of successfully/unsuccessfully retransmitted segments; and *(vi) QoE score* according to ITU-T P.1203 mode 0 [34] including extensions described in [44].[8]

$$\sum_{n} \frac{|q_n - q_{n+1}|}{q_{n+1}} \tag{7}$$

It should be noted that the startup delay is not an essential metric while comparing the performance of the ABR algorithms. It can be explained as follows. All the evaluated algorithms in this paper download the first segments with the lowest quality level to start the video as soon as possible. Therefore, the startup delay of these ABRs is approximately the same and trivial (less than 1s, compared to 300s). That means the impact of startup delay is small and unchanged among the ABRs. In spite of that, the startup delay is considered in the input file of the QoE model ITU-T P.1203. Therefore, our QoE results take into account all QoE-related parameters required by ITU-T P.1203 mode 0.

B. EXPERIMENTAL RESULTS AND ANALYSIS

In this section, we compare DoFP+ performance with state-of-the-art approaches, including ABR algorithms and the retransmission technique H2BR. Fig. 12 – 16 and Table 3 show the experimental results.

Fig. 12 compares the average bitrates of the segments in every test video for the different ABRs. AGG-H, BOLA-H, SARA-H, and BBA-0-H are the notations for the combinations of the ABRs AGG, BOLA, and SARA, and BBA-0 respectively, with H2BR. We can see that the results of an ABR vary among videos, but the difference among ABRs' performances for each video is almost unchanged. AGG downloads the lowest bitrate with around 1700kbps for all videos, followed by AGG-H with less than 10% more. The average bitrates of SARA and SARA-H are the best, with more than 3250kbps for video ToS1 and close to 3000kbps for the other videos. Our proposed method DoFP+ is nearly 2500kbps per segment over all videos, which is less by around 10% than BBA-0, BBA-0-H, and DoFP, but higher than BOLA and BOLA-H. AGG and AGG-H download the least data as the selected bitrate in AGG does not exceed the estimated throughput. With the support of H2BR, AGG-H can upgrade some low-quality segments to increase the average bitrate. SARA downloads the most data because this approach uses a high available download time for the next segment when the buffer is high, which makes it select a very high bitrate, even more than the throughput.

Video instability is reported in Fig. 13. DoFP+ provides more stability in video quality than the other ABRs except for AGG-H for all test videos. DoFP+ is able to reduce video instability by up to 77%, compared to SARA in the ToS1 video. In general, our proposed method outperforms others (except AGG-H) by from 25% to 64% in terms of video instability for all videos (see Table 3). Other ABR algorithms, even with the support of the retransmission technique H2BR, still have poor results in terms of video instability which is

[6]https://www.youtube.com/watch?v=gkIYZCmD-40
[7]https://www.youtube.com/watch?v=OQbDi3PnB2g

[8]https://github.com/Telecommunication-Telemedia-Assessment/itu-p1203-codecextension. Accessed 20 September 2021.

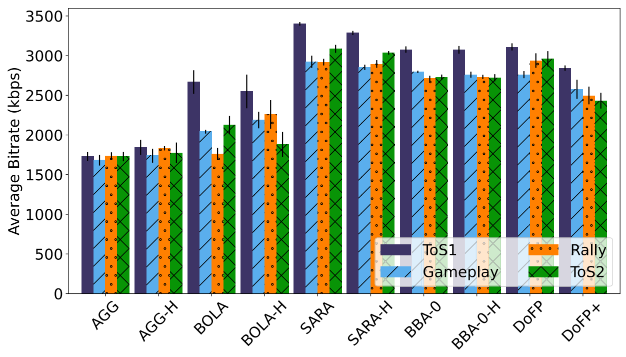


FIGURE 12. **Average bitrate.**

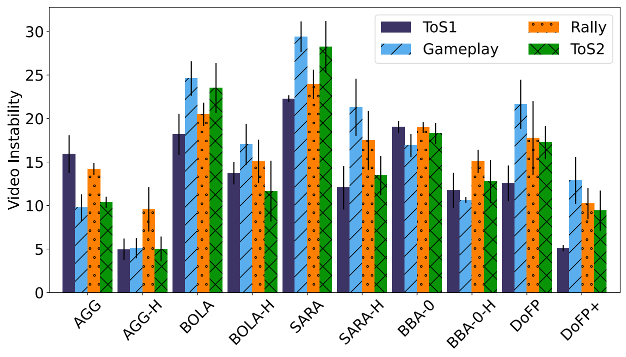


FIGURE 13. **Video instability.**

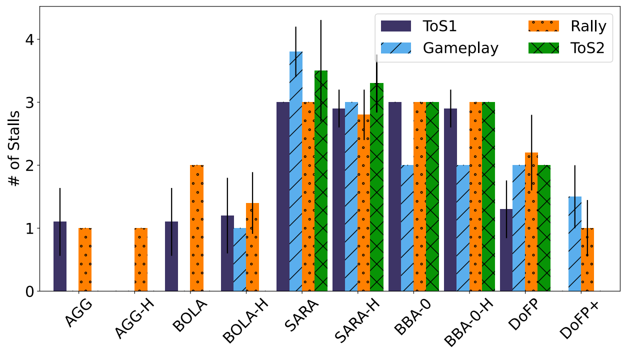


FIGURE 14. **Number of stalls.**

more than 10 in all videos. Though AGG-H provides more stable video quality, its average bitrate is dramatically less than that of DoFP+ (*e.g.*, by 33% for the Gameplay video) as shown in Fig. 12. This means that DoFP+ handles the trade-off of improving the average quality and reducing video instability quite well.

Fig. 14 and Fig. 15 provides the information about the stall events of the compared methods. Clearly, DoFP+ outperforms SARA, BBA-0 (with and without H2BR), and DoFP in both the number of stalls and stall duration. SARA and BBA-0 have, on average, three stall events in each streaming session for most cases and DoFP suffers from one stall in ToS1 and at least two stalls in other videos. On the other hand, there is no stall in the ToS1 and ToS2 videos and only

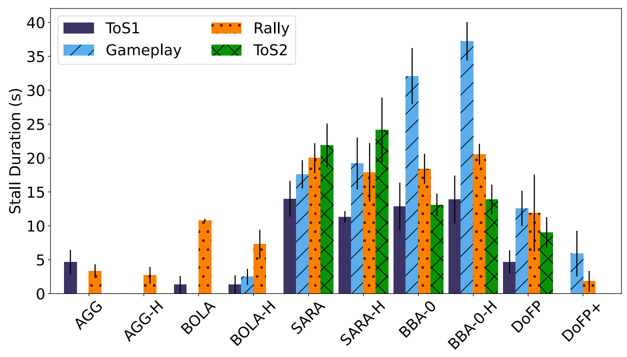


FIGURE 15. **Stall duration.**

1.5 stalls in the others when DoFP+ is deployed. AGG-H attains fewer stalls and shorter stall duration than DoFP+ due to the conservative bitrate selection. Regarding stall duration, DoFP+ also achieves comparable performance with a maximum of 5.8s of stalls. In contrast, SARA, BBA-0 (with and without H2BR), and DoFP suffer from longer stalls with a total of more than 10s (except DoFP for ToS1 and ToS2) and up to 37s. It can be seen from Table 3 that DoFP+ reduces the number of stalls and stall duration by up to 81% and 91%, respectively, across all the test videos compared with other ABR approaches excluding AGG-H.

In Fig. 15, BBA-0 (with and without H2BR) has the most severe stall problem in the Gameplay video. It can be attributed to the segment sizes. From Fig. 11, we can see that the segment sizes of this video vary significantly with the highest standard deviation (*i.e.*, std=481.2kbytes, compared to 119.8kbytes of ToS1 at bitrate 2426kbps). The BBA-0 approach considers only the bitrate level and the buffer occupancy. Thus when the size of a high bitrate segment is small (*e.g.*, segment 49 at bitrate 2426kbps), BBA-0 sees an increase in the buffer level that leads to an increase in the bitrate for the next segment (*e.g.*, segment 50). However, as this upcoming segment contains much more data and the network (at lower throughput) cannot deliver it as fast as expected, a long stall is inevitable. In contrast, the stall duration while streaming the ToS1 video is the least for most cases because this video has the most stable segment sizes (*e.g.*, std=119.8kbytes at bitrate 2426kbps).

Fig. 16 compares the QoE scores of the ABRs for all test videos. DoFP+ achieves higher QoE than the others in every video. When the ToS1 video is delivered, DoFP+ increases the QoE scores to 3.35, which is an improvement from 4% (compared to DoFP) to 29% (compared to AGG). Table 3 summarizes the analyzed results averaged over all video sequences. It lists the average percentage gain or loss (-) of DoFP+ over the compared methods for some of the considered metrics. In general, DoFP+ outperforms the compared methods and enhances the QoE by up to 13.4%, as shown in Table 3. AGG usually provides low QoE scores due to small downloaded bitrates despite fewer stall events and shorter stall durations. SARA and BBA-0 could not achieve

TABLE 3. DoFP+ performance compared to state-of-the-art approaches across all test videos. A positive number means the metric of DoFP+ is higher than that of the compared method. The text highlighted in blue and red colors represents the maximum and minimum improvement of DoFP+, respectively. Up arrows (↑) mean higher values are better and down arrows (↓) express lower values are better.

Compared Methods	Average bitrate (%) ↑	# of stalls (%) ↓	Stall duration (%) ↓	QoE score (%) ↑	Video Instability (%) ↓
AGG	**51**	19	-3	**13**	-25
AGG-H	42	**150**	**185**	9	**53**
BOLA	20	-19	-36	7	-57
BOLA-H	17	-31	-31	10	-35
SARA	**-16**	**-81**	-89	10	**-64**
SARA-H	-14	-79	-89	10	-41
BBA-0	-9	-77	-90	6	-49
BBA-0-H	-8	-77	**-91**	6	-25
DoFP	-12	-67	-80	**3**	-46

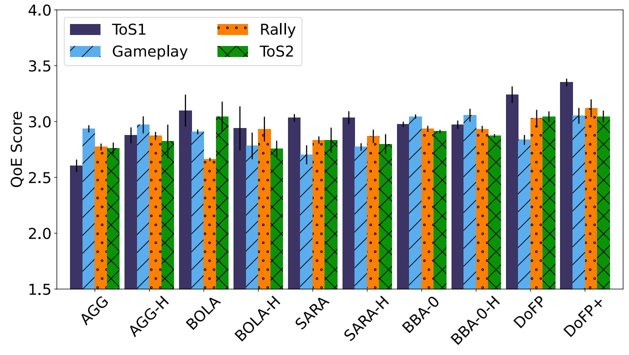


FIGURE 16. QoE score.

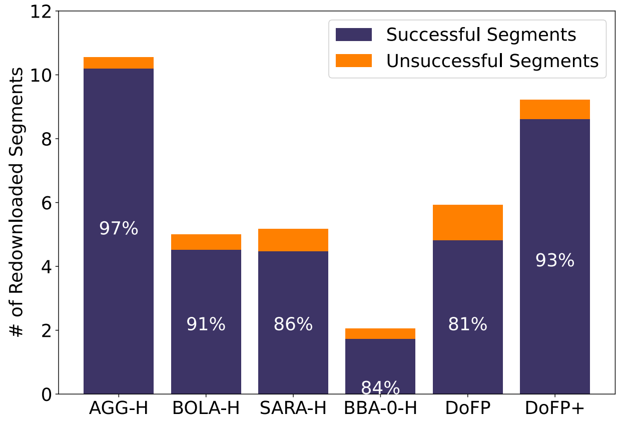


FIGURE 17. Number of redownloaded segments and percentage of successful segments.

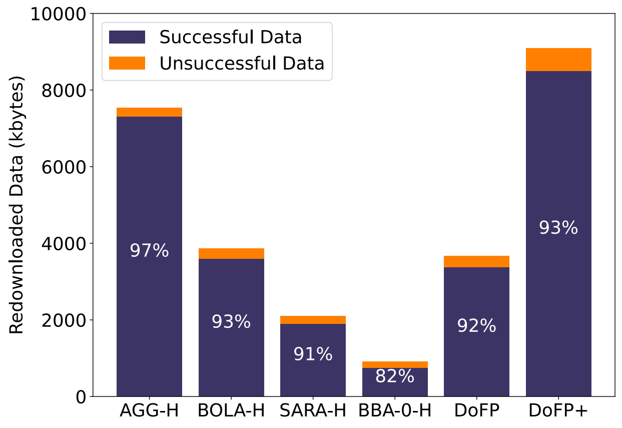


FIGURE 18. Amount of redownloaded data and percentage of successful data.

comparable results because of severe stall problems and video instability. BBA-0-H's QoE of the Gameplay video is close to that of DoFP+, but our proposed method downloads 7.5% less data according to Fig. 12. H2BR tries to upgrade the segments but the improvement is marginal. This is due to the fact that these ABR methods create quality gaps when both buffer and throughput are unfavorable, so it is difficult for H2BR to redownload segments.

The average number of redownloaded segments and the total amount of redownloaded data calculated over all videos are shown in Fig. 17 and Fig. 18, respectively. In these figures, we consider the performance of traditional ABR algorithms with the support of H2BR (*i.e.*, AGG-H, BOLA-H, SARA-H, BBA-0-H), DoFP, and DoFP+. It should be noted that successful segments/data refer to the upgraded segments/data downloaded by the client before their playout time, whereas unsuccessful segments/data cannot be played due to late arrival. DoFP+ has the second-highest number of redownloaded segments but the most redownloaded data. It is able to retransmit, on average, nearly 9 out of 75 segments of the video in a streaming session, of which 0.6 segments are unsuccessful (*i.e.*, 7% of all redownloaded segments). Those successful segments are equal to 8500 kbytes of additional downloaded data, which also means 7% of that data is unused. Meanwhile, BBA-0-H can download only two segments and 16% of them are unsuccessful. Though AGG-H is able to achieve more redownloaded segments than DoFP+, less data is additionally transferred. That means AGG-H upgrades segments with a lower bitrate than our proposed method. In addition, hybrid ABR schemes like BOLA, SARA, and buffer-based ones like BBA-0 cannot derive substantial benefits from H2BR. Though H2BR supports them to download additionally from two to five segments with up to 3900kbytes, their QoE improvement is not considerable, according to Fig. 16.

VI. CONCLUSION

In this paper, we address the research question "*What can one do with HTTP/3 to enhance HAS performance?*" by proposing an HTTP/3-based ABR algorithm that utilizes HTTP/3 features to improve the QoE of the end users. Our proposed method, DoFP+, takes advantage of *stream multiplexing*, *stream priority*, and *request cancellation* to download the next and upgraded segments that optimize an objective function. Those upgraded segments substitute lower-quality ones in the buffer to enhance the QoE of viewers. The *stream multiplexing* and *stream priority* features are used to open multiple streams at a time to download the next and upgraded segments and define the order of those segments for the purpose of optimizing QoE. The *request cancellation* feature is triggered when DoFP+ estimates some upgraded segments are expected to arrive at the client late, or the current buffer occupancy is low. The experimental results show that DoFP+ can significantly increase the QoE score while reducing video instability and stall events in all test videos, compared to a number of state-of-the-art ABR approaches. The QoE score is improved by up to 33%, and the video instability and the stall duration are decreased by up to 66% and 92%, respectively. Moreover, DoFP+ downloads more upgraded segments with higher quality levels than the state-of-the-art retransmission techniques H2BR and DoFP.

In addition, we evaluate different strategies to download retransmitted segments. We conclude that *(i)* lower-quality segments should get higher priority to be upgraded, and *(ii)* segments in the same quality gap should be downloaded sequentially in descending order of playout time with the objective of optimizing the QoE.

ACKNOWLDGMENT

(*Minh Nguyen and Daniele Lorenzi contributed equally to this work.*)

REFERENCES

[1] Cisco. *Global—2022 Forecast Highlights*. Accessed: Mar. 20, 2022. [Online]. Available: https://www.cisco.com/c/dam/m/en_us/solutions/service-provider/vni-forecast-highlights/pdf/Global_2022_Forecast_Highlights.pdf

[2] 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, Dec. 2014, doi: 10.1109/JSAC.2014.140403.

[3] A. Bentaleb, B. Taani, A. C. Begen, C. Timmerer, and R. Zimmermann, "A survey on bitrate adaptation schemes for streaming media over HTTP," *IEEE Commun. Surveys Tuts.*, vol. 21, no. 1, pp. 562–585, 1st Quart., 2019, doi: 10.1109/COMST.2018.2862938.

[4] B. Taraghi, M. Nguyen, H. Amirpour, and C. Timmerer, "INTENSE: In-depth studies on stall events and quality switches and their impact on the quality of experience in HTTP adaptive streaming," *IEEE Access*, vol. 9, pp. 118087–118098, 2021.

[5] D. Bhat, R. Deshmukh, and M. Zink, "Improving QoE of ABR streaming sessions through QUIC retransmissions," in *Proc. 26th ACM Int. Conf. Multimedia*, Oct. 2018, pp. 1616–1624, doi: 10.1145/3240508.3240664.

[6] D. Lorenzi, M. Nguyen, F. Tashtarian, S. Milani, H. Hellwagner, and C. Timmerer, "Days of future past: An optimization-based adaptive bitrate algorithm over HTTP/3," in *Proc. Workshop Evol., Perform. Interoperability QUIC*, New York, NY, USA, Dec. 2021, pp. 8–14, doi: 10.1145/3488660.3493802.

[7] M. Nguyen, C. Timmerer, and H. Hellwagner, "H2BR: An HTTP/2-based retransmission technique to improve the QoE of adaptive video streaming," in *Proc. 25th ACM Workshop Packet Video*, Jun. 2020, pp. 1–7, doi: 10.1145/3386292.3397117.

[8] M. Bishop. *Hypertext Transfer Protocol Version 3 (HTTP/3)*. Accessed: Mar. 17, 2021. [Online]. Available: https://quicwg.org/base-drafts/draft-ietf-quic-http.html

[9] J. Iyengar and M. Thomson. (May 2021). *QUIC: A UDP-Based Multiplexed and Secure Transport*. [Online]. Available: https://datatracker.ietf.org/doc/html/rfc9000

[10] M. Belshe, R. Peon, and M. Thomson. (May 2015). *Hypertext Transfer Protocol Version 2 (HTTP/2)*. [Online]. Available: https://tools.ietf.org/html/rfc7540

[11] S. Wei and V. Swaminathan, "Low latency live video streaming over HTTP 2.0," in *Proc. Netw. Operating Syst. Support Digit. Audio Video Workshop*, Mar. 2014, pp. 37–42.

[12] J. van der Hooft, C. De Boom, S. Petrangeli, T. Wauters, and F. De Turck, "Performance characterization of low-latency adaptive streaming from video portals," *IEEE Access*, vol. 6, pp. 43039–43055, 2018.

[13] P. Qian, N. Wang, and R. Tafazolli, "Achieving robust mobile web content delivery performance based on multiple coordinated QUIC connections," *IEEE Access*, vol. 6, pp. 11313–11328, 2018.

[14] M. Zverev, P. Garrido, F. Fernandez, J. Bilbao, O. Alay, S. Ferlin, A. Brunstrom, and R. Aguero, "Robust QUIC: Integrating practical coding in a low latency transport protocol," *IEEE Access*, vol. 9, pp. 138225–138244, 2021.

[15] J. Rüth, I. Poese, C. Dietzel, and O. Hohlfeld, "A first look at QUIC in the wild," in *Proc. Int. Conf. Passive Act. Netw. Meas.* Cham, Switzerland: Springer, 2018, pp. 255–268.

[16] G. Perna, M. Trevisan, D. Giordano, and I. Drago, "A first look at HTTP/3 adoption and performance," *Comput. Commun.*, vol. 187, pp. 115–124, Apr. 2022.

[17] C. Timmerer and A. Bertoni, "Advanced transport options for the dynamic adaptive streaming over HTTP," 2016, *arXiv:1606.00264*.

[18] D. Bhat, A. Rizk, and M. Zink, "Not so QUIC: A performance study of DASH over QUIC," in *Proc. 27th Workshop Netw. Operating Syst. Support Digital Audio Video*, 2017, pp. 13–18.

[19] A. Bujari, C. E. Palazzi, G. Quadrio, and D. Ronzani, "Emerging interactive applications over QUIC," in *Proc. IEEE 17th Annu. Consum. Commun. Netw. Conf. (CCNC)*, Jan. 2020, pp. 1–4.

[20] M. Nguyen, C. Timmerer, S. Pham, D. Silhavy, and A. C. Begen, "Take the red pill for H3 and see how deep the rabbit hole goes," in *Proc. 1st Mile-High Video Conf.*, New York, NY, USA, Mar. 2022, pp. 7–12, doi: 10.1145/3510450.3517302.

[21] B. Singer, *X-Men: Days Of Future Past*. Twentieth Century Fox, Los Angeles, CA, USA, 2014.

[22] K. Oku and L. Pardue. (2020). *Extensible Prioritization Scheme for HTTP*. Internet-Draft Draft-ietf-httpbis-Priority-00. [Online]. Available: https://datatracker.ietf.org/doc/html/draft-ietf-httpbis-priority-12

[23] D. V. Nguyen, H. T. Le, P. N. Nam, A. T. Pham, and T. C. Thang, "Adaptation method for video streaming over HTTP/2," *IEICE Commun. Exp.*, vol. 5, no. 3, pp. 69–73, 2016, doi: 10.1587/comex.2015XBL0177.

[24] T.-Y. Huang, R. Johari, N. McKeown, M. Trunnell, and M. Watson, "A buffer-based approach to rate adaptation: Evidence from a large video streaming service," *ACM SIGCOMM Comput. Commun. Rev.*, vol. 44, no. 4, pp. 187–198, 2015, doi: 10.1145/2619239.2626296.

[25] K. Spiteri, R. Urgaonkar, and R. K. Sitaraman, "BOLA: Near-optimal bitrate adaptation for online videos," in *Proc. 35th Annu. IEEE Int. Conf. Comput. Commun. (INFOCOM)*, Apr. 2016, pp. 1–9.

[26] P. Juluri, V. Tamarapalli, and D. Medhi, "SARA: Segment aware rate adaptation algorithm for dynamic adaptive streaming over HTTP," in *Proc. IEEE Int. Conf. Commun. Workshop (ICCW)*, Jun. 2015, pp. 1765–1770, doi: 10.1109/ICCW.2015.7247436.

[27] M. Seufert, R. Schatz, N. Wehner, B. Gardlo, and P. Casas, "Is QUIC becoming the new TCP? On the potential impact of a new protocol on networked multimedia QoE," in *Proc. 11th Int. Conf. Quality Multimedia Exper. (QoMEX)*, Jun. 2019, pp. 1–6.

[28] S. Arisu and A. C. Begen, "Quickly starting media streams using QUIC," in *Proc. 23rd Packet Video Workshop*, Jun. 2018, pp. 1–6.

[29] M. Nguyen, H. Amirpour, C. Timmerer, and H. Hellwagner, "Scalable high efficiency video coding based HTTP adaptive streaming over QUIC," in *Proc. Workshop Evol., Perform., Interoperability QUIC*, Aug. 2020, pp. 28–34, doi: 10.1145/3405796.3405829.

[30] J. Park and K. Chung, "Layer-assisted video quality adaptation for improving QoE in wireless networks," *IEEE Access*, vol. 8, pp. 77518–77527, 2020.

[31] M. Thomson and C. Benfield. *Hypertext Transfer Protocol Version 2 (HTTP/2)*. Accessed: Dec. 27, 2021. [Online]. Available: https://datatracker.ietf.org/doc/draft-ietf-httpbis-http2bis/

[32] N. Cranley, P. Perry, and L. Murphy, "User perception of adapting video quality," *Int. J. Hum.-Comput. Stud.*, vol. 64, no. 8, pp. 637–647, 2006, doi: 10.1016/j.ijhcs.2005.12.002.

[33] H. Kim and K. Chung, "Multipath-based HTTP adaptive streaming scheme for the 5G network," *IEEE Access*, vol. 8, pp. 208809–208825, 2020.

[34] *Parametric Bitstream-Based Quality Assessment of Progressive Download and Adaptive Audiovisual Streaming Services Over Reliable Transport—Video Quality Estimation Module*, Standard ITU-T Rec. P.1203. Accessed: Jul. 8, 2021. [Online]. Available: http://handle.itu.int/11.1002/ps/P1203-01

[35] W. Robitza, S. Göring, A. Raake, D. Lindegren, G. Heikkilä, J. Gustafsson, P. List, B. Feiten, U. Wüstenhagen, M. N. Garcia, and K. Yamagishi, "HTTP adaptive streaming QoE estimation with ITU-T rec, p. 1203: Open databases and software," in *Proc. 9th ACM Multimedia Syst. Conf.*, 2018, pp. 466–471, doi: 10.1145/3204949.3208124.

[36] M. Grafl and C. Timmerer, "Representation switch smoothing for adaptive HTTP streaming," in *Proc. 4th Int. Workshop Perceptual Quality Syst. (PQS)*, Sep. 2013, pp. 178–183.

[37] H. T. T. Tran, N. P. Ngoc, A. T. Pham, and T. C. Thang, "A multi-factor QoE model for adaptive streaming over mobile networks," in *Proc. IEEE Globecom Workshops (GC Wkshps)*, Dec. 2016, pp. 1–6.

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

[39] M. Nguyen, E. Cetinkaya, H. Hellwagner, and C. Timmerer, "WISH: User-centric bitrate adaptation for HTTP adaptive streaming on mobile devices," in *Proc. IEEE 23rd Int. Workshop Multimedia Signal Process. (MMSP)*, Oct. 2021, pp. 1–6.

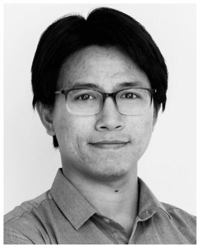
[40] D. Raca, J. J. Quinlan, A. H. Zahran, and C. J. Sreenan, "Beyond throughput: A 4G LTE dataset with channel and context metrics," in *Proc. 9th ACM Multimedia Syst. Conf.*, 2018, pp. 460–465.

[41] Y. Wang, S. Inguva, and B. Adsumilli, "YouTube UGC dataset for video compression research," in *Proc. IEEE 21st Int. Workshop Multimedia Signal Process. (MMSP)*, Sep. 2019, pp. 1–5.

[42] T. Karagkioules, D. Tsilimantos, S. Valentin, F. Wamser, B. Zeidler, M. Seufert, F. Loh, and P. Tran-Gia, "A public dataset for YouTube's mobile streaming client," in *Proc. Netw. Traffic Meas. Anal. Conf. (TMA)*, Jun. 2018, pp. 1–6.

[43] (Apr. 2020). Bitmovin Inc. *Optimal Adaptive Streaming Formats MPEG-DASH & HLS Segment Length*. Accessed: Sep. 10, 2021. [Online]. Available: https://bitmovin.com/mpeg-dash-hls-segment-length/

[44] A. Raake, M.-N. Garcia, W. Robitza, P. List, S. Goring, and B. Feiten, "A bitstream-based, scalable video-quality model for HTTP adaptive streaming: ITU-T P.1203.1," in *Proc. 9th Int. Conf. Qual. Multimedia Exper. (QoMEX)*, May 2017, p. 1203, doi: 10.1109/QoMEX.2017.7965631.



MINH NGUYEN (Member, IEEE) received the Engineering degree in electronics and telecommunications from the Hanoi University of Science and Technology, Vietnam, in 2018. He is currently pursuing the Ph.D. degree with the Institute of Information Technology (ITEC), Alpen-Adria-Universität Klagenfurt (AAU). He is also working with the Christian Doppler Laboratory ATHENA project. His research interests include adaptive video streaming, multimedia networking, computer vision, and QoS/QoE evaluation. For more information visit the link (https://minhstreaming.com).

DANIELE LORENZI (Member, IEEE) received the M.Sc. degree in ICT for internet and multimedia engineering from the University of Padua, Italy, in 2021. He is currently pursuing the Ph.D. degree with the Institute of Information Technology (ITEC), Alpen-Adria-Universität (AAU) Klagenfurt. He is also working with the Christian Doppler Laboratory ATHENA. His research interests include adaptive video streaming, immersive media, machine learning, and QoS/QoE evaluation.

FARZAD TASHTARIAN (Member, IEEE) received the Ph.D. degree in computer engineering from the Ferdowsi University of Mashhad. He is currently a Postdoctoral Researcher in the ATHENA project with the Institute of Information Technology (ITEC), Alpen-Adria-Universität Klagenfurt (AAU). Before joining the team, he was an Assistant Professor at the Azad University of Mashhad, Iran. He is a member of the Technical Program Committee of several international conferences. His current research interests include end-to-end latency and QoE in video streaming, video networking, software-defined networking, network function virtualization, mathematical modeling, and distributed optimization. For more information visit the link (https://tashtarian.net/).

HERMANN HELLWAGNER (Senior Member, IEEE) is currently a Full Professor in computer science at Alpen-Adria-Universität Klagenfurt (AAU), where he leads the Research Group Multimedia Communication (MMC), Institute of Information Technology (ITEC). He was an Associate Professor at the University of Technology of Munich (TUM) and a Senior Researcher at Siemens Corporate Research, Munich. His current research interests include distributed multimedia systems, multimedia communication and adaptation, QoS/QoE, information-centric networking, and communication in multi-UAV networks. He has published widely on parallel computer architecture, parallel programming, and multimedia communication and adaptation. He is a member of the ACM. He was a member of the Scientific Board of the Austrian Science Fund (FWF) (2005–2016) and the FWF Vice President (2013–2016). He is currently a member of the CD Senate of Christian Doppler Forschungsgesellschaft (CDG). For more information visit the link (https://www.itec.aau.at/ hellwagn/).

CHRISTIAN TIMMERER (Senior Member, IEEE) received the M.Sc. (Dipl.-Ing.) and the Ph.D. (Dr.Techn.) degrees for research on the adaptation of scalable multimedia content in streaming and constraint environments from Alpen-Adria-Universität (AAU) Klagenfurt, in January 2003 and June 2006, respectively. He is currently an Associate Professor at the Institute of Information Technology (ITEC) and the Director of the Christian Doppler Laboratory ATHENA (https://athena.itec.aau.at/). His research interests include immersive multimedia communication, streaming, adaptation, quality of experience, and sensory experience. He was the General Chair of WIAMIS 2008, QoMEX 2013, MMSys 2016, and PV 2018, and has participated in several EC-funded projects, notably DANAE, ENTHRONE, P2P-Next, ALICANTE, SocialSensor, COST IC1003 QUALINET, and ICoSOLE. He also participated in ISO/MPEG work for several years, notably in the area of MPEG-21, MPEG-M, MPEG-V, and MPEG-DASH, where he also served as the Standard Editor. In 2013, he cofounded Bitmovin (http://www.bitmovin.com/) to provide professional services around MPEG-DASH, where he was the Chief Innovation Officer (CIO)—the Head of Research and Standardization. He is a member of ACM, specifically IEEE Computer Society, IEEE Communications Society, and ACM SIGMM. He was a Guest Editor of three special issues for the IEEE Journal on Selected Areas in Communications and also served as Associate Editor for IEEE Transactions on Multimedia. For more information visit the link (http://blog.timmerer.com).