Multipath Transmission Aware ABR Algorithm for SVC HAS

Simge Gizem Ozcan, Muge Sayit*

International Computer Institute, Ege University, Izmir, Turkey

ARTICLE INFO

Keywords:
Video Streaming over HTTP
Scalable Video Coding
Adaptive Bit Rate Algorithm
Software Defined Networks
Quality of Experience

ABSTRACT

Adaptive video streaming over Hypertext Transfer Protocol (HTTP) is one of the most popular applications that have been used on the Internet in the last decade. In such applications, clients change the quality of the received video throughout the streaming session with respect to current network conditions. In these systems, the use of scalable coded video is one of the alternatives that can obtain video files encoded at different qualities. In this paper, we propose an adaptive bitrate (ABR) algorithm that aims to improve the quality of not only future segments but also the segments in the buffer. The quality of the already buffered segments is improved by downloading additional layers of the SVC-coded video, hence, preventing network resources from being wasted while increasing Quality of Experience (QoE). We use multiple paths between the server and the clients for transferring the video packets. In such systems, the ABR algorithm should determine which segment and layer should be requested from which path and when. The proposed ABR algorithm selects a group of segments to increase the quality of the buffered segments and for future segments, and selects paths for each segment in this group. Simulation results show that clients achieve higher QoE by using our approach when compared with conventional ABR algorithms. Furthermore, we show that using multiple paths cannot guarantee that QoE is maximized unless the end-system is aware of multipath transmission.

1. Introduction

Video file transmissions constitute a major share of today's Internet traffic. A recent report indicates that IP video traffic was approximately 54% in the first half of 2021 [1]. Nowadays, HTTP is one of the most preferred protocols for transferring video content through the Web due to the reliability of Transmission Control Protocol (TCP), the Hypertext Transfer Protocol (HTTP) web-cache infrastructure, and easy firewall traversal. YouTube[1] and Netflix[2] are among the most popular video streaming applications that use this protocol, and they are most responsible for increasing video traffic [2]. Almost all suggested solutions to utilize network capacity and to improve user experience follow the same approach, which enables the quality to be changed adaptively during streaming. In these systems, media content is encoded at different bitrates, which in turn leads to different qualities. In HTTP adaptive streaming systems (HAS), encoded video files with various qualities are divided into small parts called segments. This approach allows the client to make quality changes depending on the network conditions and at the same time to improve the application performance. The algorithm that decides which quality will be requested is called the adaptive bitrate (ABR) algorithm.

In HAS systems that use scalable video coding (SVC) [3], quality alternatives are constructed via a layered structure. A base layer is available for each segment, and quality improvements are provided by the enhancement layers added on the base layer. One of the main advantages of SVC is providing storage utilization due to its layered structure [4]. Recent studies show that SVC can also be used to improve the quality of the buffered segments by requesting additional enhancement layers, without removing low-quality segments and hence without wasting bandwidth resources [5], [6], [7]. In this paper, we utilize this characteristic of SVC to provide seamless streaming quality.

Enabling multipath transmission is one of the characteristics of the emerging network protocols and technologies such as Multipath TCP (MPTCP) [8], Software-Defined Networking (SDN) [9] or 5G [10] because it provides throughput improvement and better connectivity. With the use of multiple paths for the transmission of the video packets, the performance of HAS systems can be enhanced due to the increase in bandwidth. An example scenario could be streaming video from a Multi-Access Edge Computing (MEC) server to the clients in a 5G network or an SDN domain, where multipath transmission is enabled between the server and the clients. The use of MEC servers for video streaming systems is beneficial because they get the content near to the clients and help to reduce the delay. However, their storage is limited compared to the cloud servers, hence, using a scalable video codec is advantageous due to its storage utilization [4].

When SVC and multipath transmission are both utilized in HAS systems, they provide some advantages such as increased throughput and Quality of Experience (QoE) improvement. However, such a system raises the questions of which layer of which segment should be requested in which order and from which path, hence complicating the quality selection problem. In this paper, a multiple-path-aware ABR algorithm is proposed, which aims to improve the QoE of not only future segments to be requested but also the segments in the buffer. We use SDN technology to select more than one streaming path between the server and a client. These path assignments can be made by taking into account the paths' bandwidth, traffic volume, or other criteria determined by the needs of the system. SDN can help improve the QoE of HAS systems with multiple path management and routing [11].

ozcan.simgegizem@gmail.com (S.G. Ozcan); muge.sayit@ege.edu.tr (M. Sayıt)

[1] https://www.youtube.com/

[2] https://www.netflix.com/

To the best of our knowledge, this is the first study that focuses on the improvement of the quality of already-buffered segments by utilizing layered video and multipath transmission. Furthermore, the proposed ABR algorithm selects a path for each segment of each layer, a different approach from the literature. The contributions of this study can be listed as follows:

- We define an optimization model for the quality selection problem in HAS systems which uses layered video and multipath transmission. Different from the literature, the model considers quality improvement of both already buffered and future segments as well as layer dependencies. A heuristic-based ABR algorithm is proposed to solve the optimization model.
- The proposed ABR algorithm selects a group of additional layers for already buffered segments and the quality for the further segments by jointly considering the bandwidth of each path, the size of the segments and layers, the quality of the buffered segments and the buffer occupancy level.
- The ABR algorithm determines the paths to download each segment, the segments which belong to the same layer can be transmitted over different paths.
- To measure the performance, we observe each QoE parameter and the overall QoE metric. For the overall QoE metric to represent the perceived quality as accurately as possible, we determine the weights used in the QoE calculation by using a multicriteria decision-making (MCDM) method.
- We demonstrate through tests under various conditions that the use of multiple paths alone will not guarantee a performance improvement, and that appropriate approaches must be developed to benefit from this support offered by the network.

The rest of the paper is organized as follows: The background and related works are given in Second II. The proposed ABR algorithm is explained in detail in Section III. The performance of the proposed ABR algorithm is evaluated in Section IV. The conclusion is presented in Section V and the references are provided after the conclusion.

2. Background and Related Works

2.1. HAS and SVC HAS

The main challenge in HAS is how to adjust video quality according to network changes while providing a high-quality experience and eliminating video stalls. The web server usually has little knowledge of the client/network status and these systems are client-driven, i.e., all control is on the client side.

When a client starts the video streaming application and after the TCP connection with the server is established, it requests the media presentation description (MPD) file. MPD contains information about the segments of different videos kept on the server. The client then starts to request segments according to its ABR algorithm by using some criteria and the information in this file [14]. The Moving Picture Experts Group (MPEG) standardized Dynamic Adaptive Streaming over HTTP (DASH) [12], where the standard defines the MPD format and related parser functions to provide interoperability.

Many ABR algorithms have been proposed in the literature for dynamic adaptive HTTP streaming systems. These algorithms can be classified based on their input parameters, namely throughput, buffer, and hybrid throughput and buffer. In throughput-based approaches, the client may select the next segment quality on the basis of the latest [13] or average bandwidth [15]. Hybrid approaches take into account both the measured bandwidth and the buffer occupancy [16, 17]. The scalable encoded video enables clients with different network connection types, device specifications, and requests to obtain different qualities from a single encoded video file [18]. This characteristic of SVC provides storage utilization [4]. The scalable encoded video file consists of a base and several enhancement layers. There are layer dependencies in SVC. The base and lower enhancement layers are required to decode an enhancement layer. ABR algorithms developed for non-scalable video encoding HAS systems can also be used for SVC HAS. In addition, some studies focused on HAS systems that transfer scalable encoded video, where the next segment to be downloaded is selected according to the importance of the layers [19, 20].

In our previous work, we showed that downloading the layers of a buffered segment enables the improvement of the quality of segments that have not been played yet [5]. In [6], the priorities of the candidate segments to be downloaded are determined by the MCDM method and segments whose importance is above a certain level are selected for downloading by using machine learning. YouTube has the ability to remove low-quality segments from the buffer according to bandwidth conditions and request high-quality versions of the same segments instead. In [5, 21], this approach was reported to cause a waste of bandwidth. In [22], the clients run an ABR algorithm that replaces buffered low-quality segments with high-quality segments by considering the minimization of quality switches. This replacement technique is used in [23], where the retransmissions are performed via HTTP/2 features. In the next study of [23], due to the bandwidth waste caused by the replacement of low- and high-quality segments, the authors also used scalable video to minimize the quality of changes in the already buffered segments in [7]. For this purpose, the client sends one request for an enhancement layer of the oldest buffered segment.

As a different approach from the literature, in this work, we propose an approach to increase the quality of a group of buffered segments without using a retransmission mechanism and to select the additional layers to be requested under the constraint of playout times of the segments. We enhance our previous works in [6] and [5], with the usage of multipath transmission. Using multiple paths increases the number of

Table 1
Comparison of Different Studies.

Reference	Quality improvement	Layered video utilization	Multipath utilization	Path selection for each segment and layer	SDN assistance
Nguyen et al. [7]	Partial	✔	✗	N/A	✗
Kim and Chung [36], Guan et al. [38], Bentaleb et al. [24], Chen et al. [39]	✗	✗	✔	✗	✗
Cetinkaya et al. [33], Kalan and Sayit [35], Tashtarian et al. [30]	✗	✔	✔	✗	✔
Ozcan and Sayit [6]	✔	✔	✗	N/A	✔
Elgabli et al. [26]	✗	✔	✔	Partial	✗
Our work	✔	✔	✔	✔	✔

challenges, because this architecture adds more criteria to be decided: which layers of which segments should be requested, which paths should be used for each request, and when the requests should be sent. The proposed approach addresses all these selection criteria to improve QoE and to utilize network resources efficiently.

2.2. Utilizing Multiple Paths for HAS with Scalable Video

In HAS systems, the user experience can be improved by using multiple servers [24] or paths [25]; hence, the existing resources can be used more effectively. In [24], the clients request one segment from each server in each request period and the segment qualities are selected by considering buffer occupancy. In [36], the authors proposed to select a path on the basis of the recent throughput for each segment where the clients use more than one path. The authors focused on the same problem and proposed a solution based on reinforcement learning in [38]. In [39], the clients, that utilize multiple paths, request segments over different paths by minimizing the difference in arrival times of the segments. None of these works, which use multiple servers or multiple paths, utilize a layered video coding or address the quality improvement of buffered segments.

MPTCP can be an alternative to enable multipath transmission [27]. In [27], the authors show that MPTCP can be beneficial for improving the QoE of HAS systems. If end systems are aware of MPTCP, then QoE can be improved further. MPTCP scheduler may select paths for sending the segments from the server side to improve the QoE [25]. However, these solutions require changes in the transport layer [24, 26]. Also, in none of the MPTCP-related solutions the problem of path selection on the client's side for downloading the segments was studied.

Aside from MPTCP, SDN is an alternative to enable multiple paths between server and the HAS clients. In SDN technology, the control is decoupled from the forwarding elements and moved to an external device called the controller. This architecture enables SDN to manage the network traffic by making routing decisions based on flow characteristics and application requirements. Video streaming applications such as HAS can be assisted by SDN for increasing the QoE [28, 29, 9]. SDN controller might communicate with the clients [31] or dynamically re-route the flows in its domain [32] in order to direct clients about quality selection, provide network-related information, or manage the network to provide higher QoE [9].

SDN also enables one or more paths between servers and clients to be used while these paths can be assigned by considering criteria such as traffic amount or capacity of the network [33]. MPTCP and SDN can also be used together for SVC HAS. In [34], the paths used by MPTCP are selected by the SDN controller for each SVC layer in a HAS system. However, the segments of each layer are transferred over the same path; thus, path selection for the segments is not studied in [34]. Moreover, none of these studies related to SDN assistance to HAS applications focus on the client side path selection problem for segment requests and the quality improvement of the buffered segments.

Studies that use scalable video codecs and multipath transmission by utilizing SDN advantages mainly focus on streaming path selection on the network layer [33, 34], and selections are performed by the controller. Our study differs from other studies; rather than addressing the selection of streaming paths or routes on the network topology, we propose an ABR algorithm for SVC HAS and focus on selecting the paths for downloading different segments that belong to different layers. Hence, path selection is performed on the client's side. In Table 1, we provide the main differences between our proposal and the most relevant studies in the literature. All studies in the table provide quality adaptation on the client side. In the table, quality improvement refers to the quality improvement of the buffered segments. This feature is denoted as *partial* for the study in [7] because their approach focuses on only one QoE parameter, minimization of quality switches.

In the studies, an approach for layered video by using multiple paths is proposed, where each layer is routed over a different path and these path selections are managed by the SDN controller [33, 35, 30]. In our proposal, the path selec-

tion to download the segments is done on the client side, and hence different paths can be selected for each segment due to its scalable nature. In [26], the authors proposed an ABR algorithm which is aware of multiple path transmission. In the study, the layers of the requested segment are fetched from the first layer mainly and other path is used when the first link can not meet the deadline requirement [26]. Since the path selection is not done for each segment and each layer, this feature is denoted as *partial* in Table 1 for this study. Different from these similar studies in the literature, in our ABR algorithm, the clients choose a set of segments and the number of layers, and each layer of each segment can be fetched from different paths. As seen from the table, the main contribution of our work is the improvement of all QoE parameters by improving the quality of already buffered segments. The studies about path selection on the network layer are not included in the table, since they provide improvement of throughput received by the clients by making the network conditions better. In fact, these network layer related studies can easily be combined with our proposal.

By using SDN technology and SVC, the proposed ABR algorithm, which takes into account parameters such as network conditions, available bandwidth of available paths, client buffer occupancy, and layer bitrates, downloads the selected segments from appropriate paths. Furthermore, it utilizes network resources efficiently, with its approach enabling the improvement of the qualities of the buffered segments as soon as the network conditions improve without increasing rebuffering risk.

3. SVC HAS Utilizing Multipath Transmission

In the proposed system, when a client starts the application, it establishes more than one TCP connection with the server. To provide higher throughput, different routing paths can be used for the transmission of the packets. In this study, the packets that belong to different TCP connections between a server and a client over different paths are transferred thanks to SDN. In this section, we present the proposed system architecture, the functionalities of the SDN controller, and the details of the multipath transmission-aware ABR algorithm.

3.1. General Architecture of the System and Initialization of the Streaming Session

On the server side, different virtual servers that have different IPs are used to send the packets of each layer. When a client starts the video streaming application, it establishes more than one TCP connection between itself and each virtual server.

The SDN controller has network functions that provide services that measure traffic volume, track online hosts, and assign streaming paths for the flows. The traffic measurements are performed periodically and the controller calculates the available bandwidth of the paths on the basis of these measurements. On the basis of its knowledge about network topology and its service functions, when a new client starts the video application, the controller determines the set of paths for the client. When the new client sends a TCP SYN message for each TCP connection, this message is grabbed by the first hop switch and is sent to the controller. The SDN controller assigns a different streaming path on the network layer for each TCP connection by using a similar approach we used in our previous work [8]. Suppose the number of TCP connections is equal to np. The controller runs Dijkstra's algorithm np times on the basis of the available bandwidth of the links. However, the bandwidth values of the links of the selected paths should be updated in each run because when the streaming starts the video packets will be transferred over those paths and the bandwidth of the paths will change. It is not possible to estimate the bandwidth that will be used for video transmission over each path in advance. However, the average video bitrate value can be used as a good estimation for the bandwidth that will be consumed. Therefore, when the controller runs Dijkstra's algorithm, it subtracts the average video bitrate value from each link of the selected path. Then the algorithm is run again by using updated link bandwidth values. The forwarding rules related to the selected paths are sent to the switches via the OpenFlow protocol. In addition, the controller sends the available bandwidth values of the selected paths for each TCP connection to the newly joined client. If scalability problems arise due to the high number of clients in the system, the use of Server and Network Assisted DASH (SAND) based-architectures with this study is also possible. A third party server can be responsible for the communication with the clients and the SDN controller can send available bandwidth information only to that server. This approach helps to eliminate scalability problems [9, 31, 29].

3.2. Quality and Path Selection for HAS with Multipath Transmission

Considering the case where the clients are aware of multiple paths and can select not only the quality but also the path for receiving the segments, maximizing the QoE requires optimizing resources. In the selection of quality, we also consider improving the quality of the segments in the buffer. Therefore, the ABR algorithm should also make selections from buffered segments to request their higher layers. For an ABR algorithm that utilizes multiple path transmission and has these characteristics, a multi-objective optimization model is introduced in this section.

Suppose a client selects a group of segments and layers for each of them for its next request. Let G represent the number of segments within that group; pI represent the playout index, i.e., the segment being currently played; and L represent the highest layer number. The layers are numbered starting from 0. Also, let $x_{i,j}$ represent the binary variable showing that the j^{th} layer of the i^{th} segment will be requested if it equals 1 or not requested, otherwise. The objective functions of the model are given in Formulas 1-3. The first and the third objective functions aim to maximize quality and buffer occupancy, respectively. The minimum segment number that can be requested must equal pI to im-

prove the quality of already buffered segments. Equation 3 maximizes buffer occupancy by aiming to enhance the buffer horizontally, and Equation 1 maximizes the quality by aiming to enhance the buffer vertically. The second objective function aims to minimize quality changes by considering the number of layers received, which belong to consecutive segments in the buffer.

$$max \sum_{i=pI}^{pI+G} \sum_{j=1}^{L} x_{i,j} \tag{1}$$

$$min \sum_{\substack{i=pI \\ i\neq 0}}^{pI+G} \left| \sum_{j=0}^{L} x_{i,j} - \sum_{j=0}^{L} x_{i-1,j} \right| \tag{2}$$

$$max \sum_{i=pI}^{pI+G} x_{i,0} \tag{3}$$

subject to:

$$x_{i,j}.t_i^{pt} - pI < z_p.\frac{segment\_size(i,j)}{est_{thr}^{p}}, \quad \forall i \in \{pI, pI+G\}, \forall j \in \{0, L\} \tag{4}$$

$$\sum_{i=pI}^{pI+G} x_{i,j} = buffer_{size} - pI, \forall j \in \{0, L\} \tag{5}$$

$$j.x_{i,j} = \sum_{k=0}^{j} x_{i,k}, \forall i \in \{pI, pI+G\}, \forall j \in \{0, L\} \tag{6}$$

$$(i - pI).x_{i,0} = \sum_{k=pI}^{i} x_{k,0}, \forall i \in \{pI, pI+G\} \tag{7}$$

$$s_{i,j} = 0, \forall i \in \{pI, pI+G\}, \forall j \in \{0, L\} \tag{8}$$

variables:

$$x_{i,j} \in \{0,1\} \quad z_p \in \{0,1\} \tag{9}$$

The constraints of the optimization model are given in steps 4 to 8. In the equations, t_i^{pt}, $segment\_size(i,j)$, and est_{thr}^{p} represent the playout time of the i^{th} segment, the size of the j^{th} layer of the i^{th} segment, and the estimated throughput for path p, respectively. z_p is the binary variable that shows whether the path p is selected for downloading a segment and it equals 1 if it is selected for downloading the selected segment. The first constraint in the model requires the selected segment and its layer to be downloaded before its playout time by utilizing the estimated throughput of the selected path. The number of segments within a group is limited by the buffer size as determined by constraint 5. The layer dependencies are defined by constraint 6. Constraint 7 is defined to provide continuity in the downloaded segments, i.e., if i^{th} segment is received, then all previous segments should also be received. If the j^{th} layer of the i^{th} segment is received, i.e. it's in the buffer, $s_{i,j}$ equals 1. Constraint 8 is used for only requesting layers that are not in the buffer.

The selection of the segments and the layers should be completed in a limited time to have a seamless streaming session. We propose a heuristic method to solve this optimization model due to its high time complexity. In the next section, we present the details of the heuristic method.

3.3. Heuristic for Multipath Transmission-Aware ABR Algorithm

The proposed ABR algorithm considers utilizing the multiple paths and the advantages of the layered video. At the beginning of the video session, the client retrieves the bandwidth values of the paths selected by the SDN controller for each TCP connection. The client then requests the MPD file via the path with the highest bandwidth. After the MPD file is parsed, the bandwidth is re-estimated for the path from which the MPD file was retrieved. The ABR algorithm determines the segments to be requested and the paths to be used according to the estimated available bandwidth values, which will be detailed in further sections. After the requested segments are downloaded, the bandwidth values of each path are re-estimated. In this study, we used the throughput estimation method presented in DASH-JS[3], but other methods can also be used in this calculation. The formula that is used for bandwidth estimation is given in Equation 10. In the formula, bw_t refers to the bandwidth estimation value at time t and est_c is the current estimation value which is based on the throughput measured when the previous segment is being downloaded. w_1 and w_2 are the weights for previous and current bandwidth estimation values, respectively.

$$bw_t = \frac{w_1.bw_{t-1} + w_2.est_c}{w_1 + w_2} \tag{10}$$

In this study, we define the time unit that starts with receiving segments and ends with requesting the next segments as *period*. In Figure 1a the modules running on the client side are given, illustrating how a *period* starts and ends. As seen from the figure, the new *period* is not started until all requested segments are received in the previous *period*. Hence, the length of the period changes according to the current network conditions, the number of requested segments and the sizes of the requested segments. The contribution of this study, the ABR algorithm, is indicated by yellow in the figure. Figure 1b shows the general outline of the proposed ABR algorithm which consists of five steps: initialization, maximum representation selection, candidate segment group selection, path and segment selection, and backup algorithm.

The video does not start playing until the number of segments in the buffer reaches β. The time between the user clicks on the play button and the video starts playing is called *startup delay*. The playout index, which is the number of the segment being currently played by the client, increases constantly, which is why the higher layers of already buffered

[3] https://dashif.org/

segments to be requested should be carefully selected. Therefore, the ABR algorithm should determine the segments whose qualities should be improved and how much quality improvement should be provided, i.e., how many enhancement layer packets should be requested. Our studies and experiments show that a deliberate design is required because the selection strategy of these segments has a significant impact on the QoE. The base layers of future segments and the enhancement layers of already buffered segments that will be requested are determined by the ABR algorithm. To determine which segments will be suitable to be requested in the current *period*, the segments in the candidate segment group are checked prior to path selections.

The next step is path selection, i.e., selecting the paths for the transmission of each segment within the candidate segment group. After the path and segment selection step, the backup algorithm is run. In the backup algorithm, the paths that were not selected or were not totally utilized in the previous step are utilized and more base and/or enhancement layers may be requested over these paths. The details of these five steps are given in the following subsections. Table 2 gives the explanations of the variables that are used in the algorithms presented in this section.

Table 2
Definitions of the Variables Used in the Algorithms

Symbol	Description
pI	Playout index (in terms of segment number)
rep_{max}	Maximum representation quality that can be requested for the current period
$rep_{max_{prev}}$	Maximum representation quality requested for the previous period
sNo_{min}	Minimum segment number within the period
sNo_{main}	Main segment number of the period
sNo_{max}	Maximum segment number within the period
bw_{pNo}	Bandwidth of the path numbered pNo
occ_{pNo}	Path occupancy of the path numbered pNo. This value is set to 0 for all paths at the beginning of each period
occ_{buffer}	Buffer occupancy (in terms of segment count)
$size_{buffer}$	Total size of the buffer (in terms of segment count)
$size_{sNo,lNo}$	Size of the segment with segment number sNo and layer number lNo
θ	Length of a segment (in sec)
$sPath_{sNo,lNo}$	Selected path for the segment with segment number sNo and layer number lNo
$c_{sNo,lNo,pNo}$	Calculated download time of the segment with segment number sNo and layer number lNo transferred over the path numbered pNo
$cTime_{max}$	The estimated time for the completion of the set of segments within a period

3.3.1. Initialization:

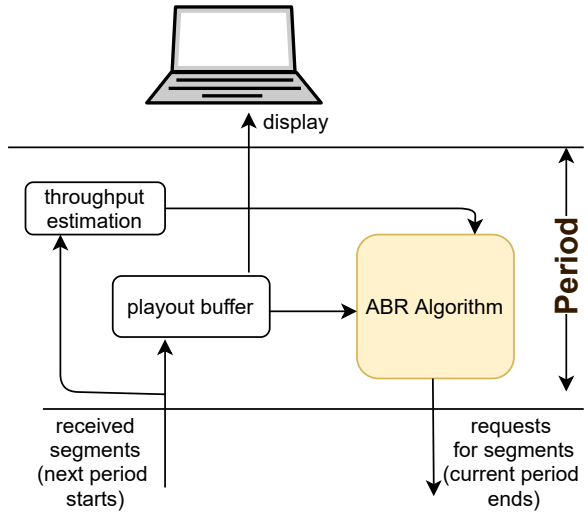
At the beginning of each *period*, the ABR algorithm runs the initialization step. In the Initialization step, the parameters specific to the current *period*, such as buffer occupancy and current available representation values are determined. The initialization step is completed when these parameters are determined. In Fig. 2, the initialization parameters in a buffer are illustrated. In the figure, BL and EL represent the base layer and the enhancement layer, respectively.The explanations and the approaches for determining these parameters are as follows:

a) Minimum segment number: The ABR algorithm selects the layers for segments within the range of the current *period*. Minimum segment number is the number of the segment with the smallest number, i.e., the segment number at the starting point of a *period*. This value (sNo_{min}) is set to $pI + \alpha$, where pI is the playout index. We do not improve the quality of the buffered segments starting directly from pI because the user continues to play the video and hence, pI constantly increases during each *period*. Here, α is used for the client to provide a safety margin which gives some time to download the segments within the candidate segment group before pI reaches sNo_{min}. Therefore, downloading a layer of a segment that has been already played is prevented. In Fig. 2, α equals 2.

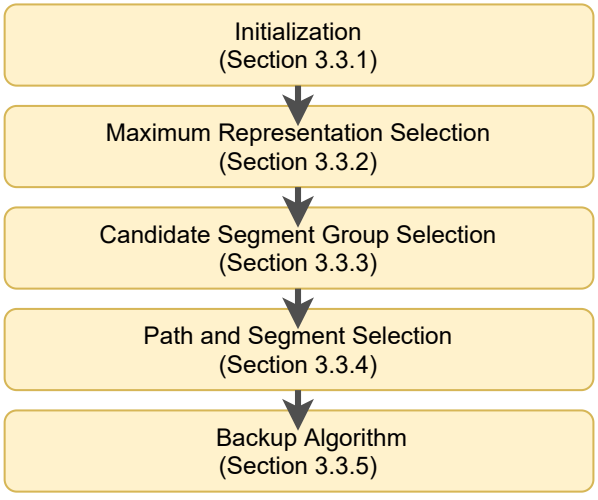
b) Maximum segment number: The maximum segment number (sNo_{Max}) value is the segment with the maximum number that can be requested from the server, considering the buffer size limit. More specifically, this value is $pI + size_{buffer}$. The calculated sNo_{max} value cannot exceed the total number of the segments of the video sequence.

c) Buffer occupancy: The buffer occupancy (occ_{buffer}) is calculated by considering the segments from pI to the first missing segment in the buffer. The buffer occupancy value is 7 for the buffer illustrated in Fig. 2. In Fig. 2, there is no gap between the downloaded segments in the buffer, however, some segments may have not arrived to the buffer in order due to the request pattern and backup algorithm given in section 3.3.5. The same calculation method is also used for the buffer occupancy in case there are gaps between downloaded segment numbers.

d) Main segment number: In the proposed ABR algorithm, the qualities of the segments whose numbers are lower than a specific segment number are improved primarily, whereas the qualities of the segments whose numbers are higher than that number are improved only in case of excessive bandwidth capacity. We call this specific segment number the main segment number. We define the main segment number to give different priority to segments according to their playout time and to download the segments on the basis of these priorities.
The clients run an algorithm to determine the main seg-



(a) Client Modules



(b) The General Outline of ABR Algorithm Steps

Figure 1: Client Modules and ABR Algorithm Steps

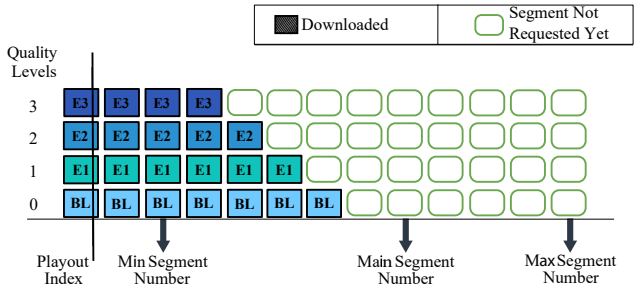


Figure 2: The Initialization Parameters

ment number, which is given in Algorithm 1. At the beginning of the video streaming session, the main segment number equals 0. In cases where the video session has just started or the maximum available representation equals the base quality level, the current period's main segment number is increased by β (the first two lines of Algorithm 1). Except for the cases given in the first line of Algorithm 1, if the buffer is not fully filled, then the main segment number is incremented by one per *period* (between the lines 3-7). Otherwise, the main segment number stays the same as the previous *period* only to improve the quality of the segment numbers whose base layer is downloaded in the previous periods.

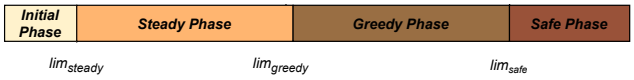


Figure 3: Illustration of Buffer Phases

Algorithm 1: Main Segment Number Setting

```
1  if sNo_main < β OR rep_max == 0 then
2  |   sNo_main ← sNo_main + β
3  else
4  |   if occ_buffer < size_buffer then
5  |   |   sNo_main ← sNo_main + 1
6  |   end
7  end
8  if sNo_main > videoLength then
9  |   sNo_main ← videoLength
10 end
```

e) Phases:
In our previous study [5], to improve the quality of the already buffered segments, only one enhancement layer was requested within a period. Hence, the clients do not send requests for more than one enhancement layer within the same period. This study showed that this approach provided a smooth quality change. In addition, in case the buffer is full enough, the increase in the number of segments to be requested keeps the buffer at a stable high level [6]. On the basis of our previous insights, we conclude that the client should act differently regarding the buffer occupancy. For this purpose, we define four different phases considering buffer fullness, namely *initial*, *steady*, *greedy*, and *safe* for this study. While the system is in the *initial* and *steady phases*, it acts more conservatively. When it is in the *greedy* and *safe phases*, more enhancement layers of various segments can be requested. The phase is used to determine the maximum available representation, i.e., the representation with maximum possible quality that can be requested, and to determine the candidate segments that will be requested within the current period. The phases are illustrated in Figure 3. The throughput estimation method differs with respect to the phase. If the phase is *initial* or *steady*, then only the bandwidth value of the path with the maximum available bandwidth is used for the throughput estimation. Otherwise, the bandwidth is calculated by summing all bandwidth values of the used paths.

3.3.2. Maximum Representation Selection:

The parameters that affect the QoE, such as current available bandwidth, segment size and buffer occupancy are highly

dynamic in HAS systems. Immediate reactions of the ABR algorithm to changes in one or more of these parameters may decrease QoE. Considering this fact, the proposed ABR algorithm uses different approaches when the system is under different circumstances to provide the maximum possible QoE.

The maximum available representation, i.e. rep_{max}, is determined by Algorithm 2 running at each $period$. This value is set by considering the throughput estimations and the buffer fullness value. rep_{max} is determined based on the bandwidth of the path with the highest available bandwidth or the sum of bandwidth values of all paths. Thus, rep_{max} defines a limit. Hence, the layers higher than rep_{max} cannot be downloaded. However, if the buffer fullness is at a good level, then requesting higher layers would not be so risky. Therefore, when the system is in the $safe\ phase$, segments of higher quality than rep_{max} can be downloaded. If the system is in the $safe\ phase$, then rep_{max} value is increased (between lines 1–3 in Algorithm 2).

The maximum representation determination algorithm aims to provide a gradual quality transition against changes in available bandwidth because significant quality changes decrease the QoE. Therefore, if the difference between the values of $rep_{max_{prev}}$ and rep_{max} of the current $period$ is more than one, then rep_{max} is increased (or decreased) incrementally to provide gradual quality changes (between lines 4–9 in Algorithm 2). Thus, the negative effect of short-time bandwidth changes on the QoE due to the significant quality switches is limited.

Algorithm 2: Maximum representation determination algorithm.

```
1  if isSafePhase then
2  |   rep_max ← rep_max + 1
3  end
4  if ABS(rep_max − rep_max_prev) > 1 then
5  |   if rep_max > rep_max_prev then
6  |   |   rep_max ← rep_max_prev + 1
7  |   else
8  |   |   rep_max ← rep_max_prev − 1
9  |   end
10 end
```

3.3.3. Candidate Segment Group Selection:

The candidate segment group represents the group of segments that are suitable to download by considering buffer and bandwidth conditions. This step of the ABR algorithm starts with determining whether a segment is suitable to be a member of the candidate segment group by using Algorithm 3. In this algorithm, all layers, which are equal or lower than rep_{max}, belonging to segments numbered between sNo_{min} and sNo_{main} are controlled at first (Algorithm 3, lines 1–2) to determine if they are appropriate to be selected. Segments are controlled whether they were requested before to prevent them from being requested again (between lines 3–5). In cases such as video session startup or buffer underrun, only the base layer of segment numbers should be downloaded to minimize startup delay or buffer underrun. Therefore, if the layer of the segment is an enhancement layer, then this segment is not added to the candidate segment group (between lines 7–9). If $lNo > 0$, the system is in the $steady\ phase$, and rep_{max} is not greater than $rep_{max_{prev}}$, then this segment also cannot be considered in the candidate segment group (between lines 10–13). These controls help keep the number of quality changes at minimum, despite a high difference in segment sizes among sequential segments and the buffer is not filled enough. In addition, removing segments from the group enables the request for new segments whose base layer has not been requested yet, thereby ensuring the system stability due to increased buffer fullness level with the newly arriving segments. Algorithm 3 outputs the candidate segment group.

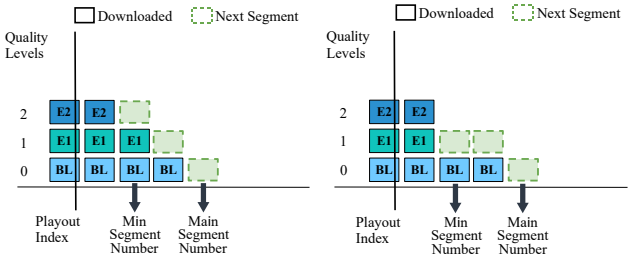
Algorithm 3: Candidate segment group selection algorithm.

```
1  foreach sNo from sNo_min to sNo_main do
2  |   foreach lNo from 0 to rep_max do
3  |   |   if isRequested(sNo, lNo) then
4  |   |   |   break;
5  |   |   end
6  |   |   if lNo > 0 then
7  |   |   |   if isInitialPhase then
8  |   |   |   |   break;
9  |   |   |   end
10 |   |   |   if isSteadyPhase AND rep_max <= rep_max_prev
   |   |   |     then
11 |   |   |   |   break;
12 |   |   |   end
13 |   |   end
14 |   |   add Segment To Candidate Group;
15 |   end
16 end
```

The control statement about the $steady\ phase$ in Algorithm 3 shows that the system phase affects the segment selection for the candidate segment group. In Figures 4a and 4b, two scenarios are given when the system is at $steady\ phase$. The segments with the highest quality and indicated as the next segment in the figure represents the rep_{max} value. Figure 4a shows the group of segments to be selected when the rep_{max} value calculated in the $period$ is the same as that of the previous $period$. If these segments are downloaded faster than the playout index increases, then the buffer fill rate will increase in the next $period$. Figure 4b illustrates the case where rep_{max} decreases when compared with the previous $period$. The group of segments to be received in the current $period$ is small. Therefore, the $period$ is completed in a short time and the algorithm can adapt to the new network conditions more quickly.

If the system is in the $greedy$ or $safe\ phase$, then it means the buffer is sufficiently filled. Hence, rep_{max} value and path selections can meet the requirements to increase QoE. Therefore, all layers up to rep_{max} of the segments are requested from the minimum segment number to the main segment number of the $period$.

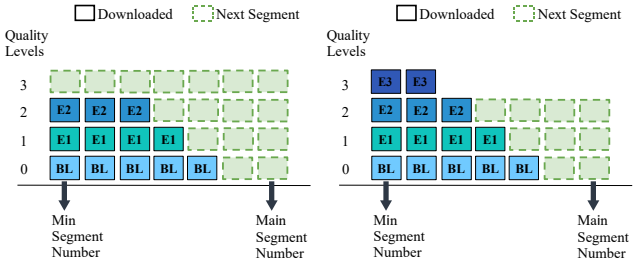
$Greedy$ and $safe\ phase$s mostly focus on improving the



(a) rep_{max} value is the same as the previous period, equals 2

(b) rep_{max} value is lower than the previous period, equals 1

Figure 4: Illustration of Candidate Segment Selection in Steady Phase



(a) rep_{max} value increases compared to previous periods

(b) rep_{max} increases to 2, after decreasing from 3 to 1

Figure 5: Illustration of Candidate Segment Selection in Greedy and Safe Phases

quality of segments whose one or more layers are received, rather than downloading new segments to maintain or increase buffer fill. In Figures 5a and 5b, two scenarios are illustrated when the system is in the $greedy$ or $safe\ phase$. Figure 5a shows the scenario, where the period's rep_{max} value increases compared with that in previous periods. Figure 5b represents another scenario where rep_{max} increases to representation 2, after decreasing from representation 3 to 1.

3.3.4. Path and Segment Selection:

The next step after determining the candidate segment group members is to select paths for downloading the segments within the group. The *period* should be completed as soon as possible. The download time of all segments via the path with the highest bandwidth might be shorter than that of some of these segments via other paths. In this case, all the segments are retrieved through the path with the highest bandwidth and the other paths are marked as unused. Unused paths are utilized by the backup algorithm, which will be explained in the next section.

Algorithm 4 determines which path will be used to receive the layer lNo of the segment numbered sNo. The inputs of the algorithm are the list of the paths in descending order in terms of bandwidths, sizes of segments from sNo_{min} to sNo_{main}, and rep_{max} value. The definitions of the symbols in the algorithm are given in Table 2.

In this algorithm, we define and utilize a variable that indicates the occupancy of the paths. occ_{pNo} shows how much time in seconds the path numbered pNo will be transferring segments. All requested segments in the previous *period* have already been downloaded at the beginning of the new *period*. Thus, no video transmission takes place in the available and usable paths. Therefore, at the beginning of each *period*, the occupancy values of all the available paths (occ_{pNo}) are set to 0 (between lines 1–3). Here, np represents the number of paths. For given sNo and lNo values, each path is checked until a suitable path is found to transfer this segment within the loop between lines 9–15 of Algorithm 4. In the 10^{th} line of the algorithm, the download time of the segment is calculated when the path numbered pNo is used. The formula $\frac{size_{sNo,lNo}}{bw_{pNo}}$ estimates how many seconds it will take to download segment with sNo and lNo layer number through the path numbered pNo. Therefore, the segment will be downloaded at $\frac{size_{sNo,lNo}}{bw_{pNo}}^{th}$ seconds after occ_{pNo} seconds passed.

Algorithm 4: The path selection algorithm.

```
1  foreach pNo from 1 to np do
2      occ_pNo ← 0.0
3  end
4  foreach sNo from sNo_min to sNo_main do
5      foreach lNo from 0 to rep_max do
6          if inCandidateGroup(sNo, lNo) then
7              minTime ← 0.0
8              sPath_sNo,lNo ← null
9              foreach pNo from 1 to np do
10                 c_sNo,lNo,pNo ← occ_pNo + size_sNo,lNo / bw_pNo
11                 if (sPath_sNo,lNo == null OR
                       minTime > c_sNo,lNo,pNo) AND
                       sNo > pI + ⌈c_sNo,lNo,pNo/θ⌉ then
12                     minTime ← c_sNo,lNo,pNo
13                     sPath_sNo,lNo ← pNo
14                 end
15             end
16             if (sPath_sNo,lNo == null) AND lNo == 0 then
17                 minTime ← c_sNo,lNo,pNo_highest
18                 sPath_sNo,lNo ← pNo_highest
19             end
20             occ_sPath_sNo,lNo ← occ_sPath_sNo,lNo + minTime
21         end
22         if (sPath_sNo,lNo != null) then
23             addSegment(sNo, lNo, sPath_sNo,lNo)
24             if isSteadyPhase then
25                 BREAK
26             end
27         end
28     end
29 end
30 cTime_max ← MAX(ALL(occ_pNo))
```

$$n - 1 < \lceil \frac{size_{sNo,lNo}}{bw_{pNo}} / \theta \rceil \tag{11}$$

If we assume that the segment duration is θ in terms of seconds, then the playout index increases by n after the segment numbered sNo and lNo is downloaded based on the inequality given in Equation 11. To ensure that the segment numbered sNo is downloaded before its playout time, the

estimated download time for each segment is calculated and checked by the algorithm whether the download time is less than the expected playout index when it is downloaded. For example, if a segment is estimated to be downloaded in 4 seconds where the segment duration (θ) is 2 seconds, then the difference between the playout index and the segment number must be greater than 2, i.e., the segment number to be requested must be greater than playout index + 2. In addition, the load of the selected path should be evaluated in the calculation by using occupancy values of the paths. These are the conditions given in the 11^{th} line of the algorithm. The path that meets the conditions is selected for downloading the segment as can be seen between lines 12–14 of Algorithm 4. In this matter, the calculated download time of this segment is added to the calculated download time of the previous segments of that *period* which are assigned to the same path (line 20). If no path is selected for a segment that belongs to the base layer, then the path with the highest bandwidth is assigned to this segment (between lines 16–19).

If the system is in the *steady phase*, then the client should download only one layer of a segment. Accordingly, the algorithm continues to search for the paths for the next segment number after selecting one layer in the *steady phase* as in the lines between 24-26 of the algorithm. More than one layer for the same segment can be requested when the system is in the *greedy phase*.

The algorithm also outputs the $cTime_{max}$ value in addition to the selected paths for downloading the segments. In the algorithm, the total estimated time to download the segments from each path is used for occupancy values of the paths. $cTime_{max}$ is the maximum value among the occupancy values of the paths. Therefore, $cTime_{max}$ gives an estimated time for the completion of the download of the segments within the related period. $cTime_{max}$ is used in the next step, by the backup algorithm.

3.3.5. Backup Algorithm:

Algorithm 5: Backup algorithm.

```
1 foreach sNo from sNo_main+1 to sNo_max do
2     foreach lNo from 0 to rep_max do
3         if c_sNo,lNo,pNo <= cTime_max then
4             occ_pNo ← occ_pNo + c_sNo,lNo,pNo
5             addSegment(sNo, lNo, pNo)
6         end
7     end
8 end
```

The last step of the ABR algorithm is the backup algorithm, which mainly creates a group of candidate segments whose number is between the main segment number and the maximum segment number for all layers without exceeding the calculated maximum representation, i.e., rep_{max}, and selects suitable paths for them. In the previous steps, the segments numbered between sNo_{min} and sNo_{main} are selected to be downloaded and path selections are made for those segments. Among the selected paths, the path with the maximum download time determines the maximum download time for the *period* ($cTime_{max}$). When segments are retrieved from a path with the maximum download time, the other paths may be idle after the segment assigned to them is transferred, which means that their bandwidths are not utilized. One of the main objectives of the proposed approach is to utilize all paths between the server and the client. For this purpose, the backup algorithm chooses new segments to be downloaded over the available paths by considering $cTime_{max}$ value. Suppose the client can download segments through three different paths. The client estimates that the segments assigned to the first path, the second path, and the third path will be downloaded within 3, 2.8, and 2.5 seconds, respectively. In this case, the second and the third paths will not be used for 0.2 and 0.5 seconds, respectively. The aim of the backup algorithm is to find appropriate segments that can be downloaded within this period of time. Considering this example, the algorithm will search for the segments that can be downloaded from the second path within 0.2 seconds and from the third path within 0.5 seconds.

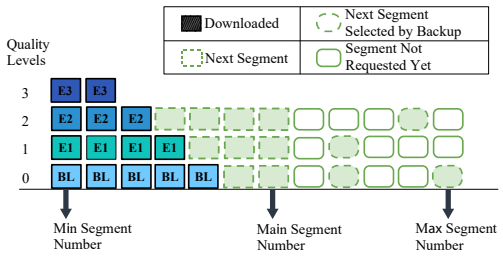


Figure 6: An Illustration of the Backup Algorithm Selection

The backup algorithm step aims to receive all the layers that can be downloaded within the current *period*, not to exceed rep_{max}, from the sNo_{main} to the sNo_{max} (Algorithm 5, line 1). The backup algorithm assigns extra segments to all other paths except the path with the calculated maximum download time. Figure 6 illustrates a possible scenario showing the selections of the segments, which are determined by the backup algorithm and to be downloaded over available paths.

4. PERFORMANCE EVALUATION

To evaluate the performance of the proposed approach, we implemented a series of tests by using the Mininet software, which is an SDN emulator running on the Linux. Floodlight [4] was used as an SDN controller in the tests.

As comparison approaches, we used two conventional ABR algorithms that request the segments of SVC encoded videos by considering throughput similar to DASH-JS. Both conventional approaches use the path with maximum available bandwidth as the main path. The conventional single-path client (convSinglePath) retrieves segments by using only this path, and the conventional multipath client (convMultiPath) improves the quality by downloading the segments of higher enhancement layers of the currently requested repre-

[4] https://floodlight.atlassian.net/wiki/spaces/HOME/overview?mode=global

Table 3
Minimum, Maximum And Average Bandwidth Values (Bw) in Mbps of the End-To-End Paths with Max Available Bandwidth in Each Scenario (Sc)

Sc	Min Bw (Mbps)	Max Bw (Mbps)	Avg Bw (Mbps)	Oscil-lations	Change Count
1	2	4	2.96	low	6
2	1	6	4.16	high	7
3	3	9	6.05	high	5
4	5	7	6.07	low	6
5	6	6	6	none	none

sentation through the additional paths. Basically, convMultiPath clients download the next higher layer of the current representation that bandwidth allows through the path with the second highest bandwidth. For the next higher layer, it uses the path among remaining paths, which has the third highest bandwidth. This selection proceeds until the client uses all paths assigned by the SDN controller. convMultiPath uses a similar approach proposed in [33]. Each type of client can select a group of segments at once and then request them. All clients' buffers keep up to 20 segments and startup delay ends after four segments have been downloaded.

In the experiments, three different videos are used, whose details are given in the next section. A total of 300 tests were performed for each video for two groups of emulations. The first group involves five single-client tests and the second group involves five multiple-clients tests per scenario. The average values of the test results are given in the graphs, which are grouped according to scenarios and all video contents. In the graphs, the confidence interval equals 95%.

4.1. Evaluation Setup

The performances of the approaches are tested over five different scenarios in which different bandwidth values and several changes in bandwidth values are implemented. The number of the paths is set to 3. Table 3 shows the average, maximum, and minimum bandwidth values, as well as the oscillation levels in the bandwidth of the main path to be used by the clients, where the emulations with one client are performed. The scenarios are set in such a way that the behaviors of the approaches can be observed in various network conditions. In the emulations with more than one client in the system, the bandwidth values given in Table 3 are multiplied by the number of clients. The bandwidth of the additional paths remains the same during the emulations. However, when more than one client transfers different layers over those paths, the available bandwidths of the additional paths change considerably during the emulation and the clients are affected by the cross traffic produced by other clients. Fig. 7 shows the actual path bandwidth value and the bandwidth observed by a client as a function of time. These values are taken from a multiple-client test with scenario 1. As seen from the figure, even if the given path bandwidth

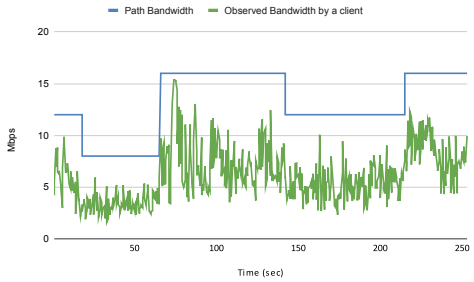


Figure 7: Path Bandwidth vs Observed Bandwidth

value remains fixed for limited intervals, the clients observe some changes in bandwidth due to the competing TCP flows.

In the experiments, we used videos with 720p resolution, namely, Big Buck Bunny (BBB), Elephants Dream (ED), and Tears of Steel (ToS) videos, which are SVC encoded according to the description given in [37] and packetized for delivery using the DASH standard. Each video has a total of four layers, including the base layer, and the segment duration is 2 seconds for all layers. Table 4 presents the main characteristics of each video content. The video durations and average bitrate for each layer are given in the table from the base layer to the highest enhancement layer. Although the bitrates of the enhancement layers are given separately in the table, the clients have to get all underlying layers to be able to decode the enhancement layers. Therefore, the playout bitrates of enhancement layers are equal to the sum of all bitrates up to that enhancement layer. These videos are selected by considering their different segment characteristics. Table 4 also shows standard deviations of the segment bitrates of each layer for each video file. In general, the size of each segment varies, and the bitrate values are the average bitrate of all segments. In Fig. 8, the segment sizes are given comparatively for each video, where the segments of each layer are grouped in the figure. These graphs show the different characteristics of the videos used in the experiments. The changes in the bitrates over time for each video file can be seen in Fig. 9. Using video files with different segment properties in terms of the variety of segment sizes allows us to achieve a wider range of performance results because we can observe the behavior of the clients by using the videos that have different characteristics in similar network conditions.

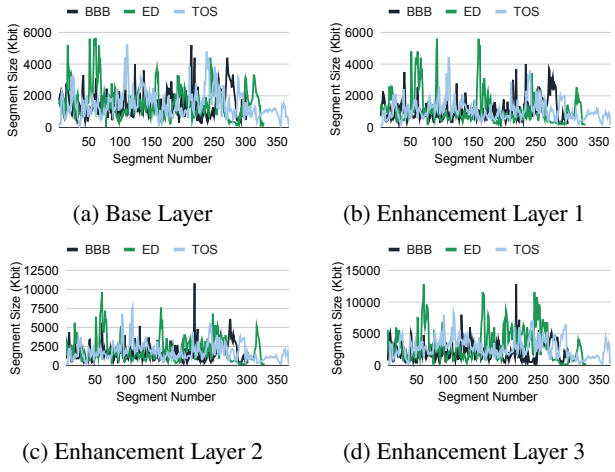
In the proposed ABR algorithm, lim_{steady}, lim_{greedy} and lim_{safe} are set to 6, 8 and 15 in terms of segment number, respectively. These values are tuned by observing the client's behavior in different conditions. The ABR algorithm code is available in [40]. It can be used in any HAS client software by importing this module as the ABR algorithm. At the beginning of the streaming, the bandwidth values of the paths should be provided to the software.

4.2. QoE Calculation

To calculate the overall QoE value, Equation (12) is used, which contains the modified version of formulas used in QoE

Table 4
Characteristics of the Video Sequences

Video	Video Duration (sec)	Bitrates from the lowest representation to the highest (Kbps)	Standard Deviations of Segment Bitrates from the lowest representation to the highest (Kbps)
Big Buck Bunny	598	1555, 1154, 1838, 2310	836, 699, 1194, 1499
Elephants Dream	654	1661, 1160, 2065, 3310	1039, 962, 1569, 2535
Tears of Steel	736	1502, 1091, 4423, 2832	853, 618, 1091, 1509



(a) Base Layer (b) Enhancement Layer 1

(c) Enhancement Layer 2 (d) Enhancement Layer 3

Figure 8: Segment Sizes per Different Video Layers

calculations [17]. "rB" refers to the average video bitrate value (in Kbps) monitored in a video session. This value is calculated by averaging the bitrate of the downloaded representations during the streaming session. A high bitrate received in a session corresponds to improve user experience. However, the received video bitrate is not the only parameter that defines the QoE on the client's side. Startup delay in terms of seconds, the number of quality changes, and rebuffering durations also negatively affect the perceived quality. Startup delay is defined by "sD" in the formula. "rC" indicates the number of quality changes in a streaming session. "uC" represents the number of buffer underruns, i.e. rebuffering events, while "uD" (in sec) is the sum of rebuffering durations in the session.

$$\begin{aligned} \forall \quad & a_0, a_1, a_2, a_3, a_4 | a_0, a_1, a_2, a_3, a_4 \in Q^+ \\ \forall \quad & rB, sD, rC, uC, uD | rB, sD, rC, uC, uD \in \{0, Q^+\} \\ & QoE^+ = rB \times a_0 \\ & QoE^- = sD \times a_1 + rC \times a_2 + uC \times a_3 + uD \times a_4 \\ & QoE = QoE^+ - QoE^- \end{aligned} \tag{12}$$

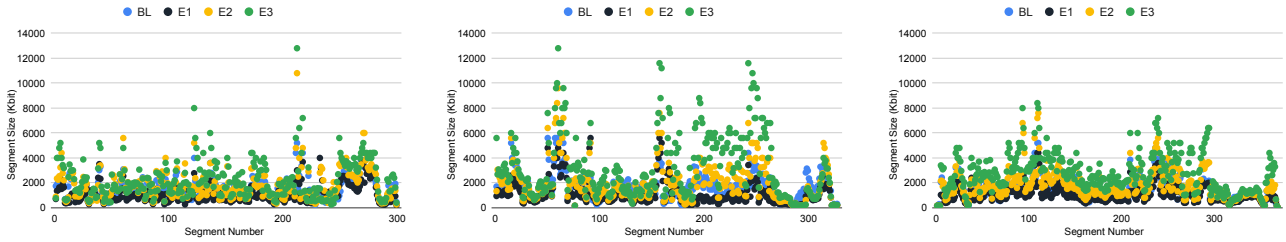
The weights from a_0 to a_4 in the formula were calculated using analytical hierarchy process (AHP), an MCDM method. MCDM is used to select the most useful option from multiple options according to the criteria determined by the decision maker. AHP is one of the popular MCDM methods because of its simplicity and robustness [6]. Basically, AHP calculates the importance of each criterion. The main objective of AHP is to determine the coefficients of the variables in a linear equation. In our case, the variables used in AHP are QoE parameters and the importance is the effects of each QoE parameter on the overall QoE. In AHP, as a first step, a decision matrix is formed by comparing the priorities of QoE parameters. In the method, each parameter used in AHP is mutually compared and values between 1 and 9 are given to these binary comparisons. These values show how much more important one parameter is than another. If the comparison value is 1, then the importance of the compared parameters is the same. If the comparison value equals 9, then the importance of the selected parameter is significantly higher than that of the compared one. In the AHP process, we mutually compared the QoE parameters and assigned values between 1 and 9 according to the information given in the studies in the literature. In dynamic adaptive systems, users prefer a gradual quality decrease rather than significant quality changes in a short time [41]. In addition, research has shown that users prefer a long rebuffering time instead of multiple ones, and they prefer quality changes and a longer startup delay to rebuffering events [42]. The importance values used in these comparisons were determined based on previous studies about the priorities of the QoE parameters. The selected weights as the output of the MCDM method are given in Table 5.

Table 5
QoE Weights

	a_0	a_1	a_2	a_3	a_4
Value	.236	.049	.092	.436	.187

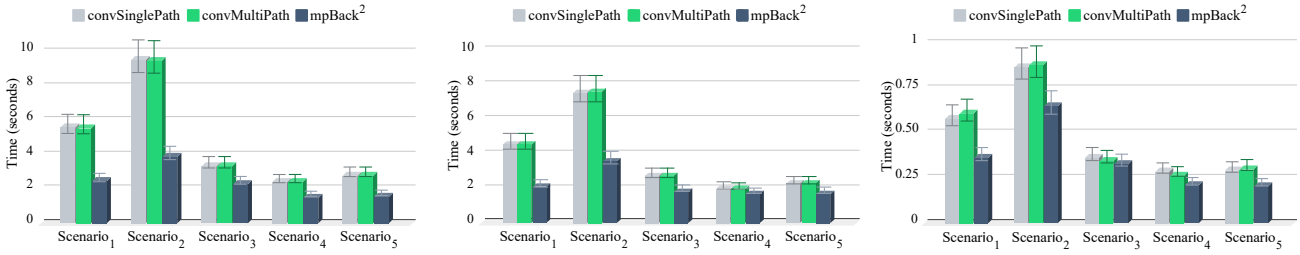
4.3. Performance Results

The test results are divided into two parts as single-client tests and multiple-clients tests. We grouped the tests in a such way because the performance of dynamic adaptive video streaming systems is affected highly when multiple-clients use the same paths due to the competing TCP flows and the reactions of the TCP congestion control algorithm in this case. The graphics are given separately for each video content. In the graphs and tables, our proposed approach is denoted as $mpBack^2$, which is due to its characteristics such as utilizing multiple paths, using a backup algorithm,



(a) Big Buck Bunny (b) Elephants Dream (c) Tears of Steel

Figure 9: Segment Sizes per Different Video Contents



(a) Big Buck Bunny (b) Elephants Dream (c) Tears of Steel

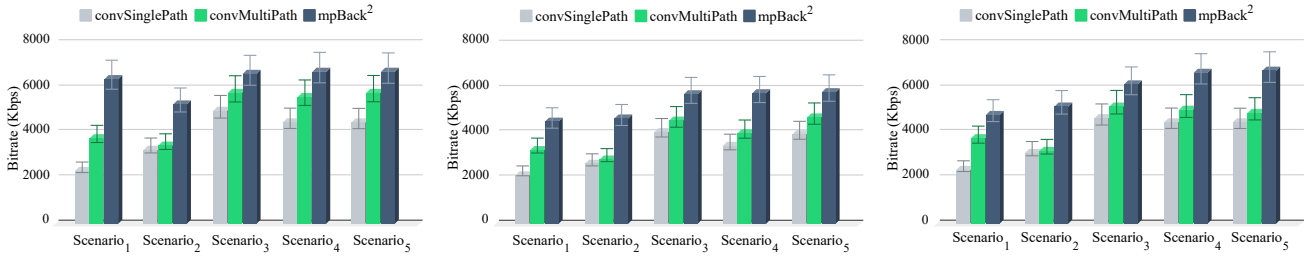
Figure 10: Average Startup Delay in Single Client Emulations (in sec)

and improving quality in two different directions, for already buffered segments and for future segments.

4.3.1. Single-Client Emulations

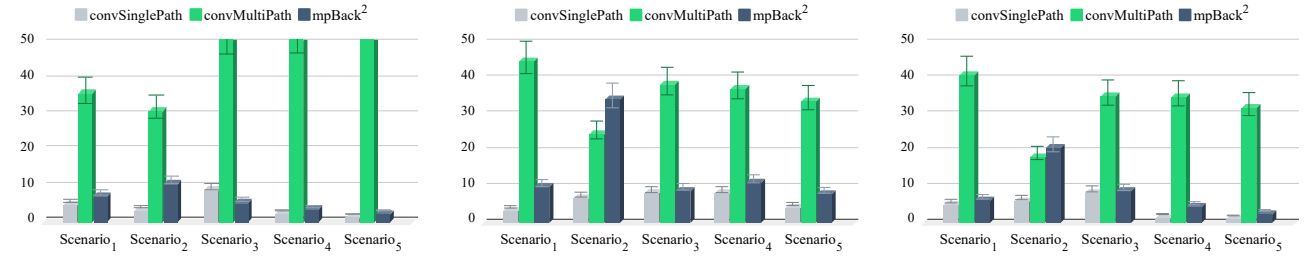
Figure 10 presents the average startup delay for each client per video content. $mpBack^2$ focuses on distributing the segments to paths to decrease the total downloading time in each period. Therefore, the proposed approach is better than the other conventional approaches in terms of startup delay, especially for the BBB and ED video sequences. The startup delay values of $mpBack^2$ are even less than those of observed with convMultiPath approach. This result shows the importance and effectiveness of the arrangement of which segments should be requested over which paths when multiple paths are utilized. The startup delay values are lower for the ToS video file than for other video files because the size of the first several segments is too low in this video.

The average received representation bitrates are shown in Figure 11. For all scenarios and each video content, the average representation bitrates in the proposed approach are higher than those in the other approaches. This finding shows that the proposed algorithm provides better visual quality under these circumstances. When compared with the convMultiPath, this improvement was measured as a maximum of 69% and an average of 36%. Nonetheless, a higher received bitrate does not always mean higher QoE. In Figure 12, the average number of representation changes is comparatively given for all types of clients. This value obtained in



(a) Big Buck Bunny (b) Elephants Dream (c) Tears of Steel

Figure 11: Average Representation Bitrates in Single Client Emulations



(a) Big Buck Bunny (b) Elephants Dream (c) Tears of Steel

Figure 12: Average Number of Representation Changes in Single Client Emulations

the proposed approach is less than that of the convMultiPath for most scenarios of all video content. In the results of the second scenario, given in Figures 12b and 12c, more representation changes are observed than conventional clients because a gradual transition is preferred by the proposed approach rather than a sharp quality change. For example, suppose the video has n layers. The quality may suddenly jump to the highest from the lowest for the two consecutive requests with the other ABR algorithms. The proposed approach increases the quality within $n-1$ step incrementally in this case. Hence, the number of quality changes seems high in scenarios such as 2, where the number of changes in bandwidth values is high.

The distribution of the requested qualities in the video sessions is given in Figure 13. The clients running the proposed approach are able to play the highest quality segments in all tests. In scenarios 1 and 2 of all video contents, both conventional approaches cannot play any segment from the highest quality, while the percentage of segments that have the highest quality is a maximum of 88% and average of 45% for the proposed approach. These results show that the clients using the $mpBack^2$ approach *i*) successfully utilized the estimated throughput when multiple paths are used and *ii*) manage to maintain a high buffer occupancy level during the whole streaming session.

In the proposed ABR algorithm, the clients minimize the number of buffer underruns by controlling buffer fullness and by sending requests for lower-layer segments when the measured throughput decreases. Table 6 presents statistics about buffer underruns for all scenarios for all video contents. As shown in the tables, with the proposed approach, buffer underruns are observed only in the second scenario, where the video file is ToS and in the first two scenarios with the ED video. Buffer underruns are observed in most scenarios for all video content with the conventional approaches where the bandwidth is dynamically changed. Considering that users prefer slightly increased startup delay or representation changes instead of buffer underruns, the higher representation changes of $mpBack^2$ in the second scenarios of ED and ToS are compensated by the decrease in the buffer underrun duration.

Regarding the calculated QoE values (Figure 14), the proposed approach provides a clear improvement in all single-client emulations compared with the other clients. Considering that these QoE values represent the most important metrics related to perceptual quality, the proposed approach has achieved its purpose.

Table 6
Average Number of Buffer Underruns and Durations (sec) in Single-Client Emulations

	Single Conv		Multi Conv		Benchmark		$mpBack^2$	
Sc	#	Time	#	Time	#	Time	#	Time
1	1	11.4	1	9.9	0	0	0	0
2	.8	4	.8	4.65	.5	3.32	0	0
3	1	9.2	1	10.56	0	0	0	0
4	0	0	.2	2.5	0	0	0	0
5	0	0	0	0	0	0	0	0

(a) Big Buck Bunny

	Single Conv		Multi Conv		Benchmark		$mpBack^2$	
Sc	#	Time	#	Time	#	Time	#	Time
1	2.2	40.95	2.2	40.96	2	5.78	1.8	.92
2	2.2	34.27	2.2	36.01	.9	4.25	.2	.07
3	0	0	0	0	0	0	0	0
4	1	11.23	1	11.85	0	0	0	0
5	2.8	29.08	3	30.88	1.5	3.85	0	0

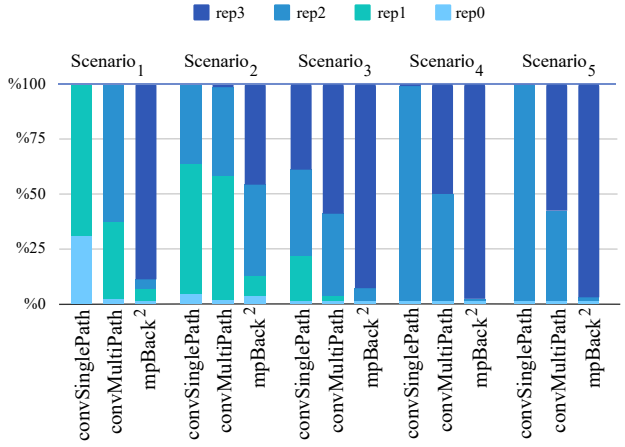
(b) Elephants Dream

	Single Conv		Multi Conv		Benchmark		$mpBack^2$	
Sc	#	Time	#	Time	#	Time	#	Time
1	0	0	0	0	0	0	0	0
2	.2	1.56	0	0	.2	.8	.2	.11
3	0	0	.8	1.32	0	0	0	0
4	0	0	.2	5.15	0	0	0	0
5	0	0	0	0	0	0	0	0

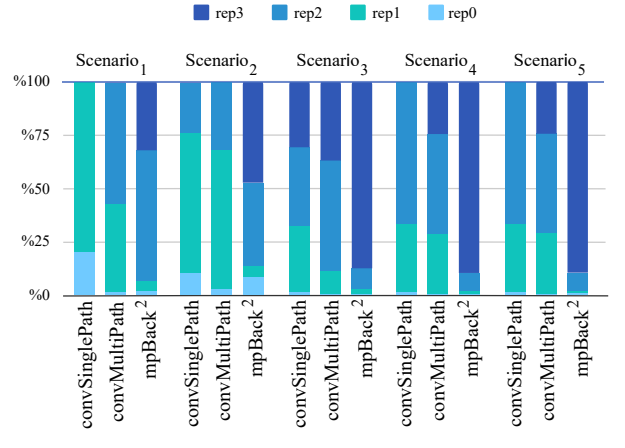
(c) Tears of Steel

4.3.2. Multiple-Client Emulations

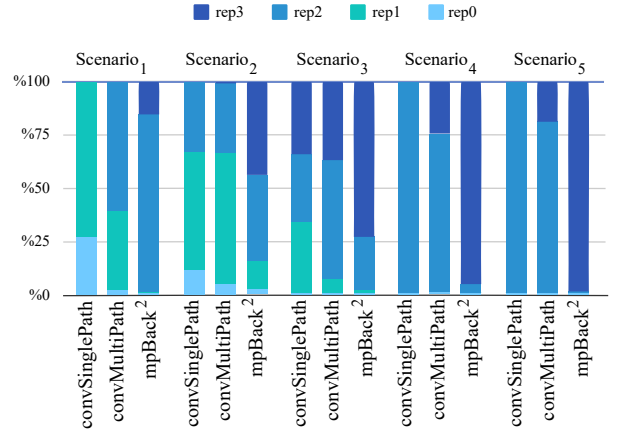
In the tests with multiple online clients, four clients were connected simultaneously to the network: convSinglePath, convMultiPath, and two $mpBack^2$ clients. In the tables and graphics, the two $mpBack^2$ clients are labeled as $mpBack^2_1$ and $mpBack^2_2$. Multiple-client emulations were performed for comparison with conventional approaches and for ob-



(a) Big Buck Bunny



(b) Elephants Dream



(c) Tears of Steel

Figure 13: Requested Quality Distribution in Single Client Emulations

serving the performance where more than one client that runs the proposed ABR algorithm share the same network.

Figure 15 shows that the proposed approach decreased the startup delay values in multiple-client emulations. Figure 16 shows the average representation bitrate values observed in the video sessions. For multiple-client tests, the

Table 7
Average Number of Buffer Underruns snd Durations (in sec) in Multiple-Client Emulations

	Single Conv		Multi Conv		$mpBack^2_1$		$mpBack^2_2$	
Sc	#	Time	#	Time	#	Time	#	Time
1	3.4	42.72	4	49.08	0	0	0	0
2	4	58.62	4.2	61.76	1.4	4.44	1.4	7.5
3	1.4	12.88	1.2	10.26	0.4	0.3	0.2	0.2
4	0	0	0	0	0	0	0	0
5	0.2	0.8	0	0	0	0	0	0

(a) Big Buck Bunny

	Single Conv		Multi Conv		$mpBack^2_1$		$mpBack^2_2$	
Sc	#	Time	#	Time	#	Time	#	Time
1	8.4	131.44	10	148	1.4	1.6	1.4	1
2	5.6	66.76	7.2	88.12	2.4	2	2.4	3.19
3	3	33.56	3.2	38.12	0	0	0	0
4	3	43.60	2.4	39.5	0	0	0	0
5	3.8	47.49	4.4	54	0	0	0	0

(b) Elephants Dream

	Single Conv		Multi Conv		$mpBack^2_1$		$mpBack^2_2$	
Sc	#	Time	#	Time	#	Time	#	Time
1	7.6	8.7	9.2	112.1	0	0	0	0
2	8.2	111.1	9.4	125.8	3.4	3.48	3.2	2.97
3	0.4	2.61	1	6.86	0	0	0	0
4	1.6	15.38	1.8	18.4	0	0	0	0
5	1	7.19	1.2	9.99	0	0	0	0

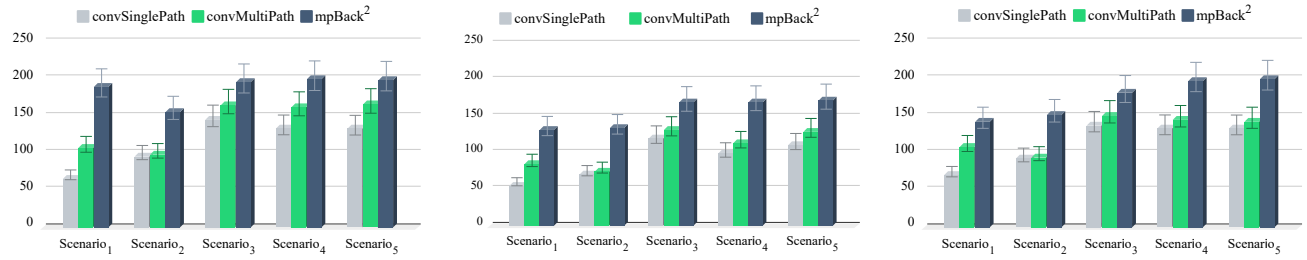
(c) Tears of Steel

bandwidths of all paths increased in direct proportion to the number of clients. As a result, more capacity is provided to the clients, and each client can use the available bandwidth. This setting enables the clients to receive the highest quality for all videos in scenarios 3, 4 and 5. For this reason, the representation values of all clients for these scenarios are close to each other and all approaches are able to play these videos mostly with the highest quality.

Figure 17 shows the average number of representation changes observed in the streaming sessions. The reason for the small variation in the number of representation changes in the BBB scenarios 4 and 5 is that the available bandwidth is generally sufficient to download the representation with the highest quality for most of the session. Hence, the representation quality remains the same for most of the segments. In other scenarios where the clients compete for bandwidth, a maximum improvement of 95% was measured compared with two conventional clients. The average improvement rates are 79% higher compared with convSinglePath and 75% higher compared with convMultiPath.

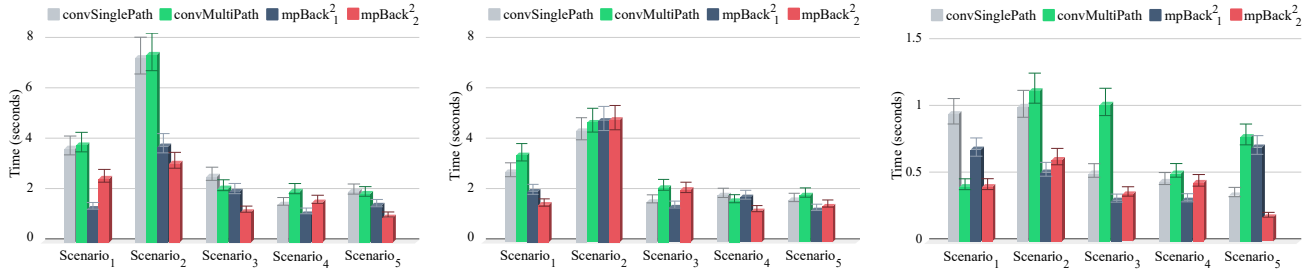
Figure 18 presents the distribution of the qualities of the requested segments as stack graphs. Similar to the improvement in the received representation bitrate values, the quality of the requested segments improved due to the usage of multiple paths and the estimation based on total bandwidth.

The multiple-client emulations showed a significant dif-



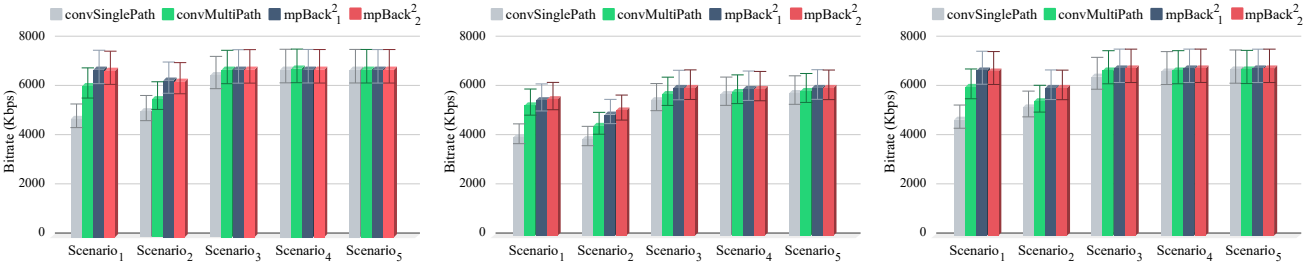
(a) Big Buck Bunny (b) Elephants Dream (c) Tears of Steel

Figure 14: Average QoE in Single Client Emulations



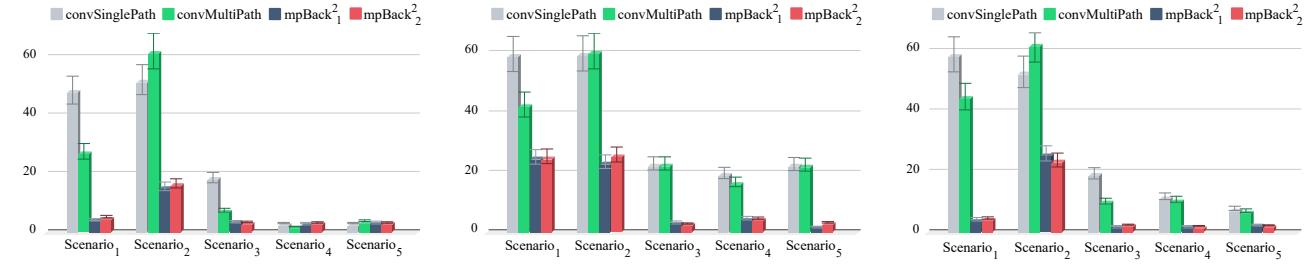
(a) Big Buck Bunny (b) Elephants Dream (c) Tears of Steel

Figure 15: Average Initial Waiting Times in Multiple-Client Emulations



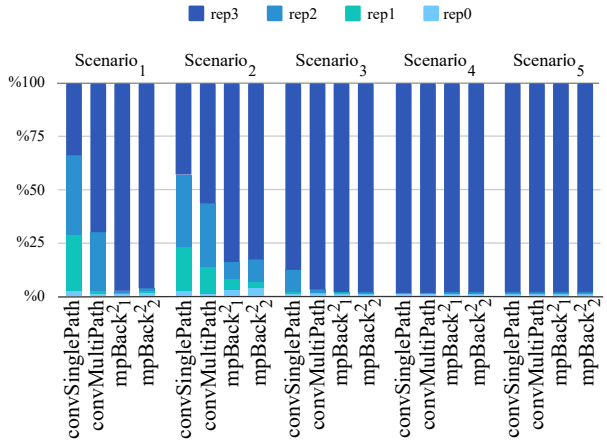
(a) Big Buck Bunny (b) Elephants Dream (c) Tears of Steel

Figure 16: Average Representation Bitrates in Multiple-Client Emulations

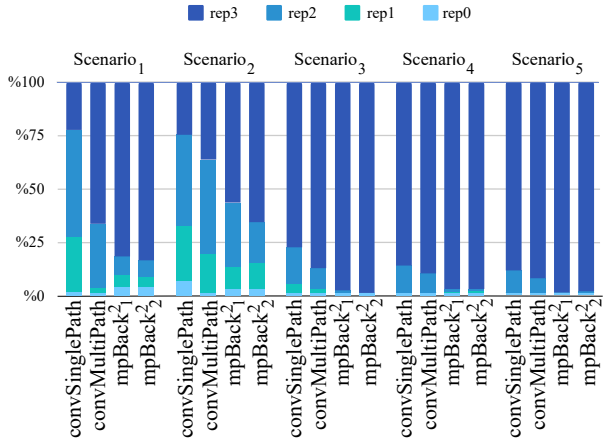


(a) Big Buck Bunny (b) Elephants Dream (c) Tears of Steel

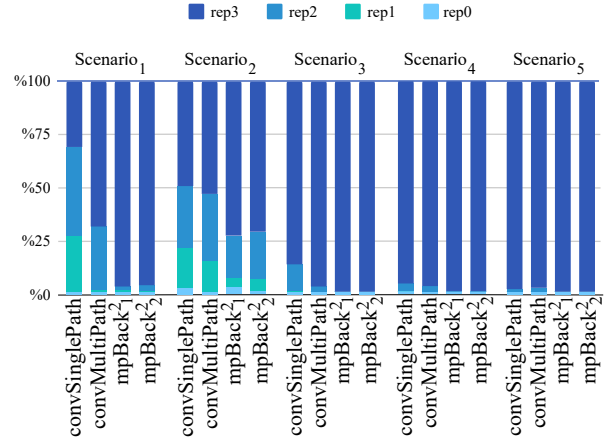
Figure 17: Average Number of Representation Changes in Multiple-Client Emulations



(a) Big Buck Bunny



(b) Elephants Dream



(c) Tears of Steel

Figure 18: Requested Quality Distribution in Multiple-Client Emulations

ference in the number and the duration of buffer underruns, where the performance of the $mpBack^2$ approach is compared with that of single-client emulations (Table 7). Both conventional clients experience buffer underruns in all tests except scenario 4 of BBB and elongated buffer underrun durations compared with the single-client tests. Even though the number of representation changes and the startup delay in the proposed approach are greater than those in conventional approaches in some scenarios, the overall QoE is still higher in $mpBack^2$ because buffer underruns are limited in these scenarios, where conventional clients suffer from a higher number and duration of underruns. In the proposed approach, the number of underruns was 83% better than that of the convSinglePath and up to 86% better than that of convMultiPath. Buffer underrun duration values with the proposed approach are up to 95% lower than that of both conventional approaches.

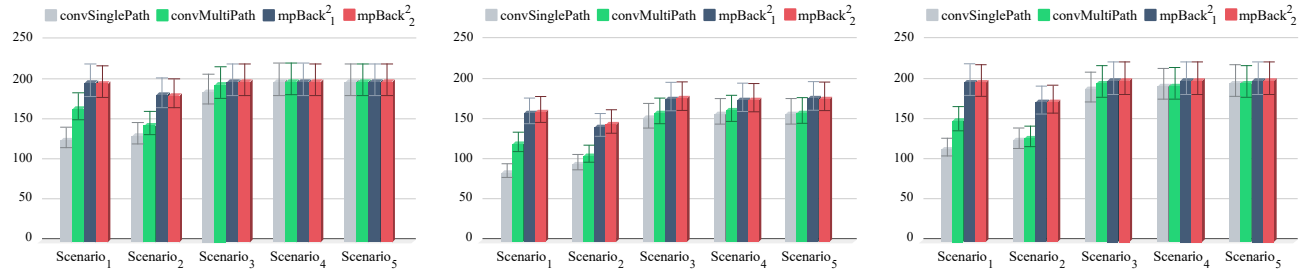
In all experiments, we used SVC encoded videos. If H.264 AVC encoded videos were used in the comparison approaches and the bandwidth was fixed and limited so that only the base layer can be downloaded during the whole streaming session, then the comparison approaches would outperform our approach because of the bitrate overhead of the H.264 SVC base layer and the lack of the flexibility to improve the quality of the buffered segments.However, if the bandwidth is dynamic and if higher quality segments are transmitted after low quality video is buffered with AVC coded video, then this approach would consume more bandwidth than the transmission of enhancement layers after the base layer of SVC.

5. Conclusion

Increased throughput is the main factor that helps improve the performance of video streaming systems such as Hypertext Transfer Protocol (HTTP) Adaptive Streaming. Such an increase can be achieved by increasing the available bandwidth. Using more than one path to transfer video packets between server and client is one of the preferred approaches. In this paper, we proposed an Adaptive Bitrate (ABR) algorithm for HAS, which utilizes Scalable Video Coding (SVC) and multipath communication. The main characteristic of the ABR algorithm is that it improves the quality of the already buffered segments without wasting network resources, because of SVC's features.

Clearly, using more than one path helps increase throughput in end-to-end communication. However, when multiple paths are used in a HTTP Adaptive Streaming (HAS) system, the Quality of Experience (QoE) might be negatively affected due to the high number of changes in throughput. Our observations and conducted experiments with the conventional approach show that, to eliminate the negative effects of the multiple path usage on QoE and to optimally use resources, HAS applications should be aware of the multipath transmission and the ABR algorithm should be designed carefully by considering the effects of multipath transmission. Two conventional approaches using single and multiple paths were used as comparison approaches in the emulations.

Evaluation results show that using only SVC or multiple paths without a well-planned strategy is not enough to maximize QoE. The direct positive effects on QoE of multipath transmission can be seen by comparing the perfor-



(a) Big Buck Bunny (b) Elephants Dream (c) Tears of Steel

Figure 19: Average QoE in Multiple-Client Emulations

mances of convSinglePath and convMultiPath approaches. The simulation results obtained under various network settings show that using more than one path increases the overall quality while providing a decrease in buffer underruns as expected. However, the tests also show that the observed increase in video quality due to the usage of multiple paths is limited unless additional methods that utilize multipath transmission are used at the end system. Accordingly, the proposed algorithm increased the values of the positive QoE metrics while minimizing the negative ones when compared with both conventional clients, using single and multipath transmission. The QoE metrics are used to evaluate the overall QoE value of the session, which utilizes an multicriteria decision-making (MCDM) method considering the impacts of each QoE metric. The proposed approach improved the QoE up to 80% compared with a client which is not aware of multipath transmission.

The increase in the quality provided by the proposed approach compared with the other applications is caused not only by the use of multiple paths but also by the techniques used to improve the already-buffered segments by utilizing the advantages of SVC. Even though HAS systems can cope with sudden network changes successfully, once the segments are buffered with a certain quality, the only way to improve the quality of the buffered segments is to re-download them. Instead of this basic approach which wastes time and bandwidth, the proposed approach of increasing the quality of buffered segments by using SVC layers shows that higher QoE can be obtained under sudden bandwidth changes. Hence, the adaptivity to dynamic network conditions in HAS systems may not be limited to the current segments but it can also be provided for past segments.

Dynamic buffer size under different video properties and network conditions, as well as machine learning-based estimation for available maximum representation will be studied in future work. Multipath TCP (MPTCP) can also be used for multipath communication. Developing an MPTCP scheduler that considers HAS characteristics and works with the proposed system is another future plan.

Acknowledgment

This work is funded by the Scientific and Technological Research Council of Turkey (TUBITAK) Electric, Electronic and Informatics Research Group (EEEAG) under grant 115E449. We are grateful to Ege University Planning and Monitoring Coordination of Organizational Development and Directorate of Library and Documentation for their support in editing and proofreading service of this study.

References

[1] Sandvine, Global Internet Phenomena Report 2022 [Online]. Accessed: August 30, 2022 https://www.sandvine.com/phenomena

[2] J. Jiang, V. Sekar, and H. Zhang, 2014. Improving Fairness, Efficiency, and Stability in HTTP-Based Adaptive Video Streaming With Festive, IEEE/ACM Trans. on Netw. 22, 326–340.

[3] H. Schwarz, D. Marpe and T. Wiegand, 2007. Overview of the Scalable Video Coding Extension of the H.264/AVC Standard, IEEE Transactions on Circuits and Systems for Video Technology, 17(9), 1103-1120.

[4] Y. Sánchez, C. Hellge, T. Schierl, W. Van Leekwijck, Y. Le Louédec and F. Orange-FT, 2011. Scalable Video Coding based DASH for Efficient Usage of Network Resources, W3C Web and TV Work., 19-20.

[5] S.G. Ozcan, T. Kivilcim, C. Cetinkaya, and M. Sayit, 2017. Rate Adaptation Algorithm with Backward Quality Increasing Property for SVC-DASH, IEEE 7th International Conference on Consumer Electronics - Berlin (ICCE-Berlin), Berlin, 24–28.

[6] S.G. Ozcan and M. Sayit, 2019. Improving the QoE of DASH over SDN: A MCDM Method with an Intelligent Approach, 22nd Conference on Innovation in Clouds, Internet and Networks and Workshops (ICIN), 100–105.

[7] M. Nguyen, H. Amirpour, C. Timmerer, and H Hellwagner, 2020. Scalable High Efficiency Video Coding based HTTP Adaptive Streaming over QUIC, Workshop on the Evolution, Performance, and Interoperability of QUIC (EPIQ '20), Association for Computing Machinery, New York, NY, USA, 28–34.

[8] K. Herguner, R. S. Kalan, C. Cetinkaya, and M. Sayit, 2017. Towards QoS-aware Routing for DASH Utilizing MPTCP over SDN. IEEE Conference on Network Function Virtualization and Software Defined Networks (NFV-SDN), Berlin, 1-6.

[9] A.A. Barakabitze, N. Barman, A. Ahmad, S. Zadtootaghaj, L. Sun, M.G. Martini, L. Atzori, 2020. QoE Management of Multimedia Streaming Services in Future Networks: A Tutorial and Survey, IEEE Communications Surveys & Tutorials, 22(1), 526-565.

[10] G. Ghatak, Y. Sharma, K. Zaid, A. U. Rahman, 2020. Elastic Multi-connectivity in 5G Networks, Physical Communication, 43, 101176.

[11] M. Ramakrishna, R.C. Fernandes, and A.K. Karunakar, 2017. Estimation of Adaptation Parameters for Scalable Video Streaming

over Software Defined Networks, Procedia Computer Science, 115, 715–722.

[12] T. Stockhammer, 2011. Dynamic Adaptive Streaming over HTTP : Standards and Design Principles, Second Annual ACM Conference on Multimedia Systems, ACM, New York, NY, USA, 133–144.

[13] B. Rainer, S. Lederer, C. Müller, C. Timmerer, 2012. A Seamless Web Integration of Adaptive HTTP streaming, 20th European Signal Processing Conference (EUSIPCO), 1519–1523.

[14] A. Bentaleb, B. Taani, A. C. Begen, C. Timmerer and R. Zimmermann, 2019. A Survey on Bitrate Adaptation Schemes for Streaming Media Over HTTP, IEEE Communications Surveys & Tutorials, 21(1): 562-585.

[15] G. Cofano, L.D. Cicco, T. Zinner, A. Nguyen-Ngoc, P. Tran-Gia, and S. Mascolo, 2017. Design and Performance Evaluation of Network-assisted Control Strategies for HTTP Adaptive Streaming, ACM Trans. on Mult. Comput. Commun. Appl., 13(3s), 42:1–24.

[16] S. Petrangeli, J. Famaey, M. Claeys, S. Latré, and F.De Turck, 2015. QoE-Driven Rate Adaptation Heuristic for Fair Adaptive Video Streaming, ACM Trans. on Multimedia Comput. Commun. Appl. 12, 28:1-24.

[17] X. Yin, A. Jindal, V. Sekar, and B. Sinopoli, 2015. A Control-Theoretic Approach for Dynamic Adaptive Video Streaming over HTTP, SIGCOMM '15, ACM, New York, NY, USA, 325–338.

[18] X. Lu, G.R. Martin, 2013. Improved Rate Control Algorithm for Scalable Video Coding, Schloss Dagstuhl - Leibniz-Zentrum fuer Informatik GmbH, Wadern/Saarbruecken, Germany.

[19] S. Amir Hosseini, Z. Lu, G.D. Veciana, S.S. Panwari 2016. SVC-Based Multi-User Streamloading for Wireless Networks, IEEE Journal on Selected Areas in Communications, 34, 2185–2197.

[20] J. Hwang, J. Lee, and C. Yoo, 2016. Eliminating Bandwidth Estimation from Adaptive Video Streaming in Wireless Networks, Image Commun., 47, 242–251.

[21] A. Mondal, S. Sengupta, B.R. Reddy, M. J.V. Koundinya, C. Govindarajan, P. De, N. Ganguly, and S. Chakraborty, 2017. Candid with YouTube: Adaptive Streaming Behavior and Implications on Data Consumption, 27th Workshop on Network and Operating Systems Support for Digital Audio and Video (NOSSDAV'17), ACM, New York, NY, USA, 19–24.

[22] C. Wang, A. Rizk, and M. Zink, 2016. SQUAD: A Spectrum-based Quality Adaptation for Dynamic Adaptive Streaming over HTTP, 7th International Conference on Multimedia Systems. Association for Computing Machinery, New York, NY, USA, Article 1, 1–12.

[23] M. Nguyen, C. Timmerer, and H. Hellwagner, 2020. H2BR: An HTTP/2-based Retransmission Technique to Improve the QoE of Adaptive Video Streaming, 25th ACM Workshop on Packet Video, ACM, New York, NY, USA, 1–7.

[24] A. Bentaleb, P.K. Yadav, W.T. Ooi, and R. Zimmermann, 2020. DQ-DASH: A Queuing Theory Approach to Distributed Adaptive Video Streaming, ACM Trans. Multimedia Comput. Commun. Appl., 16, 1, Article 4, 24 pages.

[25] B. Han, F. Qian, L. Ji, and V. Gopalakrishnan, 2016. MP-DASH: Adaptive Video Streaming Over Preference-Aware Multipath, CoNEXT '16, ACM, USA, 129–143.

[26] A. Elgabli, K. Liu, V. Aggarwal, 2019. Optimized Preference-Aware Multi-path Video Streaming with Scalable Video Coding, IEEE Transactions on Mobile Computing, 1–1.

[27] H.K. Yarnagula, R. Anandi, V. Tamarapalli, 2019. Objective QoE Assessment of DASH Adaptation Algorithms over Multipath TCP, 11th International Conference on Communication Systems & Networks (COMSNETS), Bengaluru, India, 461-464.

[28] R.S. Kalan, M. Sayit and A.C. Begen, 2018. Implementation of SAND Architecture Using SDN, 2018 IEEE NFV-SDN, Italy, 1-6.

[29] A. Bentaleb, A.C. Begen, R. Zimmermann and S. Harous, 2017. SDNHAS: An SDN-Enabled Architecture to Optimize QoE in HTTP Adaptive Streaming, IEEE Transactions on Multimedia, 19(10), 2136-2151.

[30] F. Tashtarian, A. Erfanian, and A. Varasteh, 2018. S2VC: An SDN-based Framework for Maximizing QoE in SVC-Based HTTP Adaptive Streaming, Computer Networks, 146, 33–46.

[31] I. M. Ozcelik and C. Ersoy, 2019. Chunk Duration-Aware SDN-Assisted DASH. ACM Trans. Multimedia Comput. Commun. Appl., 15(3), 22 pages.

[32] C. Cetinkaya, K. Herguner, C. Hellge, M. Sayit, 2020. Segment-aware Dynamic Routing for DASH Flows over Software-Defined Networks, International Journal of Network Management, 30(4),e2102.

[33] C. Cetinkaya, Y. Ozveren, and M. Sayit, 2015. An SDN-assisted System Design for Improving Performance of SVC-DASH, Federated Conference on Computer Science and Information Systems (FedCSIS), 819–826.

[34] A. Gohar, A. Raza, S. Lee, 2019. Dynamic Adaptive Streaming Over HTTP Using Scalable Video Coding with Multipath TCP in SDN, International Conference on Information Networking (ICOIN), 480–484.

[35] R. S. Kalan and M. Sayit, 2021. SDN Assisted Codec, Path and Quality Selection for HTTP Adaptive Streaming, IEEE Access, 9, 129917-129932.

[36] H. Kim and K. Chung, 2019. Segment Scheduling Scheme for Efficient Bandwidth Utilization of HTTP Adaptive Streaming in Multipath Environments, IEEE Access, 7, 36910-36920.

[37] C. Kreuzberger, D. Posch, H. Hellwagner, 2015. A Scalable Video Coding Dataset and Toolchain for Dynamic Adaptive Streaming over HTTP, 6th ACM Multimedia Systems Conference (MMSys), 213–218.

[38] Y. Guan, Y. Zhang, B. Wang, K. Bian, X. Xiong and L. Song, 2020. PERM: Neural Adaptive Video Streaming with Multi-path Transmission, IEEE INFOCOM 2020 - IEEE Conference on Computer Communications, 1103-1112.

[39] Y. Chen, D. Towsley and R. Khalili, 2016. MSPlayer: Multi-Source and Multi-Path Video Streaming, IEEE Journal on Selected Areas in Communications, 34(8), 2198-2206.

[40] $mpBack^2$ Rate Adaptation Algorithm Code. Available at https://gist.github.com/SimgeGizemO/544dd21e4c120fb05f20d5d8c2839ea3.

[41] R.K.P. Mok, X. Luo, E.W.W. Chan, and R.K.C. Chang, 2012. QDASH: A QoE-aware DASH System, 3rd Multimedia Systems Conference, ACM, New York, NY, USA, 11–22.

[42] M. Seufert, S. Egger, M. Slanina, T. Zinner, T. Hoßfeld, P. Tran-Gia, 2015. A Survey on Quality of Experience of HTTP Adaptive Streaming, IEEE Communications Surveys Tutorials, 17, 469–492.