ES-HAS: An Edge- and SDN-Assisted Framework for HTTP Adaptive Video Streaming

Reza Farahani, Farzad Tashtarian, Alireza Erfanian, Christian Timmerer, Mohammad Ghanbari * and Hermann Hellwagner

Christian Doppler Laboratory ATHENA, Alpen-Adria-Universität, Klagenfurt, Austria
*School of Computer Science and Electronic Engineering, University of Essex, Colchester, UK

Abstract

Recently, *HTTP Adaptive Streaming* (HAS) has become the dominant video delivery technology over the Internet. In HAS, clients have full control over the media streaming and adaptation processes. Lack of coordination among the clients and lack of awareness of the network conditions may lead to sub-optimal user experience and resource utilization in a pure client-based HAS adaptation scheme. *Software Defined Networking* (SDN) has recently been considered to enhance the video streaming process. In this paper, we leverage the capability of SDN and *Network Function Virtualization* (NFV) to introduce an edge- and SDN-assisted video streaming framework called ES-HAS. We employ virtualized edge components to collect HAS clients' requests and retrieve networking information in a time-slotted manner. These components then perform an optimization model in a time-slotted manner to efficiently serve clients' requests by selecting an optimal cache server (with the shortest fetch time). In case of a cache miss, a client's request is served (*i*) by an optimal replacement quality (only better quality levels with minimum deviation) from a cache server, or (*ii*) by the original requested quality level from the origin server. This approach is validated through experiments on a large-scale testbed, and the performance of our framework is compared to pure client-based strategies and the SABR system [12]. Although SABR and ES-HAS show (almost) identical performance in the number of quality switches, ES-HAS outperforms SABR in terms of playback bitrate and the number of stalls by at least 70% and 40%, respectively.

CCS Concepts

• **Information systems** → **Information systems applications**;
• **Multimedia information systems** → *Multimedia streaming*.

Keywords

Dynamic Adaptive Streaming over HTTP (DASH), Edge Computing, Network-Assisted Video Streaming, Quality of Experience (QoE), Software Defined Networking (SDN), Network Function Virtualization (NFV).

ACM Reference Format:
Reza Farahani, Farzad Tashtarian, Alireza Erfanian, Christian Timmerer, Mohammad Ghanbari and Hermann Hellwagner. 2021. ES-HAS: An Edge- and SDN-Assisted Framework for HTTP Adaptive Video Streaming. In *Workshop on Network and Operating System Support for Digital Audio and Video (NOSSDAV '21)) (NOSSDAV '21), September 28-October 1 2021, Istanbul, Turkey*. ACM, New York, NY, USA, 7 pages. https://doi.org/10.1145/3458306.3460997

NOSSDAV '21, September 28-October 1 2021, Istanbul, Turkey

ACM ISBN 978-1-4503-8435-3/21/09…$15.00
https://doi.org/10.1145/3458306.3460997

1 Introduction

Over the last few years, video traffic has become the dominant type of traffic over the Internet and is expected to reach more than 80% of the total IP traffic by 2022 [14]. Employing HTTP on top of TCP has risen recently for video transmission over the Internet. Most video streaming services employ *HTTP Adaptive Streaming* (HAS) like *Dynamic Adaptive Streaming over HTTP* (DASH) [28], one of the standardized delivery methods for video streaming. In DASH, each video is divided into segments of a given duration, *e.g.*, between 2 and 10 seconds. Each segment is encoded into several different quality levels based on bitrate or resolution (*i.e.*, representations). This information, including the segments' locations, *e.g.*, cache server or origin server, is stored within the Media Presentation Description (MPD). The users' *Quality of Experience* (QoE) is significantly affected by the video quality levels that the video players will select. DASH provides the capability that video players adapt the video quality by considering the client's resources, *e.g.*, playout buffer and/or the current network conditions. The adaptation process can be performed with different schemes categorized into *(i)* pure client-based, *(ii)* client-based assisted by network components, and *(iii)* network-assisted. When DASH clients receive the requested MPD in a pure client-based adaptation scheme, they run their local adaptation logic to decide about the next segment quality. The decision based on the local parameters, *e.g.*, buffer status and estimated available bandwidth, can lead to a sub-optimal approach due to insufficient information about the network. In the second approach, the clients' adaptation decision is assisted by a network component like a proxy server. In the network-assisted method, the adaptation decision is performed via a centralized network component with a global view of the entire network topology [11]. Thus, the network-assisted approach can be more beneficial for the users' QoE. Fundamental paradigms of modern networks, *i.e.*, *Software Defined Networking* (SDN) and *Network Function Virtualization* (NFV) have been used in modern network-assisted frameworks [7, 8, 12, 18, 24]. The control plane and data plane are decoupled in the SDN approach, and network intelligence is placed in a component called SDN controller[22]. This controller has a complete view of the network by obtaining information from each network component. The SDN controller can thus manage the network efficiently [22]. As a complementary technology to SDN, NFV provides the ability to virtualize different traditional network functions as *Virtual Network Functions* (VNFs) [13]. Therefore, SDN and

NFV can be utilized to improve network performance, management, and Quality of Service (QoS) parameters, or can be used for a particular goal, *e.g.*, increasing users' QoE for video streams. In this paper, we propose ES-HAS as an Edge- and SDN-Assisted Framework for HTTP Adaptive Video Streaming. In our framework, an edge component called *Virtual Reverse Proxy Server* (VRP) is introduced to assist clients in receiving requested quality levels from the optimal cache servers (with shortest fetch time). In case of a cache miss, a client's request is served (*i*) by an optimal replacement quality (only better quality levels with minimum deviation) from a cache server, or (*ii*) by the original requested quality level from the origin server. This goal is achieved through a comprehensive network view provided by the SDN controller, collecting relevant information from the CDN (cache server occupancy) and from clients (requested video qualities), in a time-slotted manner. ES-HAS adopts and extends the core idea of [12] to introduce a new network-assisted video streaming framework. ES-HAS improves user satisfaction without any modification on the client-side. Although we use the standard DASH, ES-HAS is not limited to DASH, and it can be extended to other HAS formats. The main contributions of this paper can be summarized as follows: *(i)* We leverage the SDN and NFV concepts and design an architecture in three different layers to efficiently assist DASH clients' adaptation processes. *(ii)* We propose a mixed-integer linear programming (MILP) model to serve clients' requests from optimal cache servers. The model determines a replacement quality instead of the original requested quality, in case that quality is not available in the cache servers, with minimum quality deviation. *(iii)* We analyze our proposed framework performance through a series of experiments conducted in the CloudLab [27] environment on a large-scale testbed and compare it with [12] and the pure client-based adaptation approach.

The remainder of the paper is organized as follows. Section 2 reviews related work. We motivate our work through an example in Section 3 before elaborating on the details of the proposed approach and optimization model in Section 4. The evaluation setup and results are presented in Section 5. Finally, Section 6 concludes the paper and gives a few directions of the future work.

2 Related Work

In most prior works, the mechanism for adaptation is implemented within the player. However, it has been shown that this approach encounters instability and difficulty in ensuring bandwidth fairness, especially in a shared network environment [6, 19]. Several client-based adaptation strategies were proposed to improve users' QoE by considering the aforementioned problems [20, 25, 26]. However, some approaches like FESTIVE [20] are vulnerable to instability when the number of HAS players increases in a shared network, possibly due to a bandwidth overestimation result, plus require significant modifications on the client-side [9]. Network elements can be utilized to assist DASH players in deciding the next segment's representation. A server-client cooperation approach named ESTC was presented in [16]. It uses two independent algorithms on the server side and client side to achieve adaptive video streaming fairness, efficiency, and stability. Although this strategy employs network element(s) to improve the users' QoE, the next representation's adaptation decision is still made individually by clients. Bentaleb *et al.* [9] used a bandwidth estimation method in SDNDASH to allocate network resources for improving QoE per client that was extended in SDNHAS [10] to support a cluster-based estimation for bandwidth requests. Server and Network-Assisted Dynamic Adaptive Streaming over HTTP (SAND) [30] was introduced by MPEG and tries to mitigate DASH performance problems by enabling protocol messages to be exchanged among network components, *e.g.*, CDN servers. However, SAND does not specify how to handle such messages efficiently. SAND defines four message types, *i.e.*, status messages, metrics messages like buffer occupancy, packets enhancing reception, and packets enhancing delivery. If a network component can process all mentioned messages or a subset of them, it is called a DASH-aware network element (DANE). Although these systems can improve client-side adaptation decisions, they are not straightforward to implement, and only a few papers have pursued this approach yet. The authors of [12] designed an SDN-enabled network-assisted framework for HAS systems, entitled SABR. SABR collects various information items from the network side such as available bandwidth and cache occupancy to guide player bitrate decisions. Erfanian *et al.* [17] leveraged the SDN and NFV concepts and introduced a cost-aware real-time video streaming approach. It uses a set of the virtualized components at the edge of the network to collect data from the client side, cooperate with the SDN controller, transcode video streams, and deliver them through a hybrid multicast/unicast approach. In this paper, we use NFV and SDN technologies to introduce ES-HAS. ES-HAS collects information from both network side and client side and employs virtual reverse proxy (VRP) servers at the edge of an SDN-enabled network to provide network assistance for HAS clients in case of both cache hits and misses.

3 Motivating Example

We present our main motivation by means of the following example. Concerning the components involved in the example, we refer to the ES-HAS architecture depicted in Fig. 1 and the fact that ES-HAS adopts and extends the core idea of the SABR framework [12]. We consider a simplistic scenario with only one cache server, an SDN controller, several OpenFlow (OF) switches, one VRP, and two clients. We assume clients request their next video segments in an SABR-enabled system [12] (Fig. 2(a)) and an ES-HAS system (Fig. 2(b)). As illustrated, the SDN controller should initially receive *cache map* messages from cache servers (step 1). Each cache map indicates the presence of representations (quality levels) of requested video segments. The structure of a cache map is illustrated in Fig. 1. Whenever an OF CDN-side switch cannot find a matching rule for (1), it sends a *packet-in* message (2) through the OF protocol to the SDN controller. The SDN controller replies to the packet-in (3) and installs a path for the cache server. Finally, the cache map will be received by the SDN controller (4).

In an SABR-based system (Fig. 2(a)), the clients send requests (5) to the SDN controller for acquiring cache map and network status information. An OF client-side switch uses OF packet-in messages (6) for these non-matching HTTP requests. After obtaining the replies (7), the OF switch forwards these requests to the SDN controller (8). Assuming that the clients support SABR, after receiving the requested information from the SDN controller (9), they determine a joint cache map, including all cache servers for the requested segments. The desired segments are requested

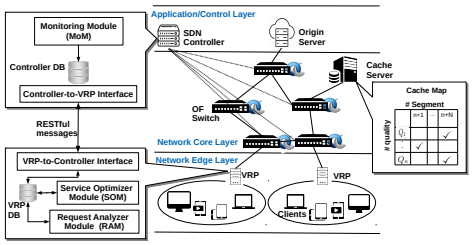

Figure 1: Proposed ES-HAS architecture

by the clients (10 − 11). Then, the SDN controller installs the corresponding routes between the clients and the determined cache servers (12 − 13). Consequently, the requested segments (14) are transferred to the clients (15). It is clear that when the number of DASH clients increases, the number of exchanged messages to/from the SDN controller (via OF and HTTP) will increase proportionally. Hence, system efficiency will decrease gradually. Our proposed framework employs a VRP at the edge of the network to overcome the aforementioned problem. A VRP works in time slots as follows. As shown in Fig. 2(b), the clients send requests (5) to the VRP for the desired segments' qualities, and the VRP collects these received requests in each time slot. The VRP plays the role of a gateway for the client to the network and vice versa. Therefore, the VRP requests the cache map and network status information from the SDN controller for all collected requests (6 − 10). Using aggregate messages in these steps decreases the number of exchanged messages to/from the SDN controller (via OF and HTTP) as compared to SABR. After receiving the demanded information from the SDN controller (10), the VRP runs an optimization program to determine the optimal cache servers for the gathered clients' requests. For the sake of simplicity, let us assume that the requested segment quality levels are available on the cache servers. Then they are requested from the cache servers and transmitted to the VRP (11 − 16). A second reduction of exchanged messages (as compared to SABR) may occur in this stage since the VRP does not forward identical requests by clients redundantly to the SDN controller and cache servers. Finally, the fetched segments are transferred to the clients (17). The details of the ES-HAS framework will be discussed in the next sections. The number of communicated messages to/from the SDN controller for both systems (Fig. 2) is shown in Table 1. In real scenarios with a large video dataset, this amount of exchanged messages could overload the SDN controller in an SABR-enabled system. In our proposed structure, we introduce VRPs at the edge of the system for collecting client-side requests, which increases costs and imposes additional delay to the system. However, the number of messages to/from the SDN controller, the load on the SDN controller, and the network bandwidth consumption are significantly reduced. With an increased number of clients, we can enlarge the VRP's resources or, alternatively, add another VRP to manage more clients efficiently.

4 System Model and Problem Formulation

ES-HAS has three main layers: *(i)* Application/Control, *(ii)* Network Core, and *(iii)* Network Edge (Fig. 1). On the application/control layer, the SDN controller periodically monitors available bandwidth and the cache servers' occupancy information (cache maps) and stores them in its database. Thus, we define a *Monitoring Module* (MoM) as the controller's main application module to collect the aforementioned information from the OpenFlow switches. Therefore, the database in the controller has accurate information about cache servers and paths' available bandwidths and serves the VRPs' requests through the *Controller-to-VRP Interface*.

Table 1: Number of messages to/from SDN controller

Arch.	Number of . . . OF msgs.	. . . HTTP msgs.
SABR	$2N_{cache} + 5N_{client}$	$N_{cache} + 2N_{client}$
ES-HAS	$2N_{cache} + 2N_{VRP} + 3N_{cache}N_{VRP}$	$N_{cache} + 2N_{VRP}$

The network core layer consists of the OpenFlow switches connected to the SDN controller, CDN components including an origin server, and multiple cache servers. In the network edge layer, we employ several VRPs, which are equipped with three main modules as follows: *(i) Request Analyzer Module* (RAM), *(ii) Service Optimizer Module* (SOM), and *(iii) VRP-to-Controller Interface*. Before describing the modules, it is noted that VRPs operate in time slots with an equal duration of θ. As shown in Fig. 3, each time slot consists of two intervals: *(i)* Data Collecting and *(ii)* Optimization interval. In the first interval, users' requests are gathered and aggregated by a VRP's RAM. To prevent sending identical requests (issued by multiple clients in a given time slot), RAM identifies them and considers only one request per segment. Moreover, using RESTful messages, the VRP periodically retrieves the required information (cache maps plus available bandwidths between each cache server and the VRP) from the SDN controller and stores them in its DB. Then, during the first interval, the VRP can fetch them from its database. The SOM is executed by the VRP in the second interval to serve clients' requests optimally. Note that, according to the SOM's results, the data transmission will start in the first interval of the next time slot. Actually, for each request, using the optimization model presented later in this section, the SOM selects an appropriate cache server that hosts the requested quality. However, when the requested quality is not available in a cache server, the SOM either determines an optimal replacement quality level on a cache or fetches the originally requested quality from the origin server. In this paper, we employ the concept that a *replacement quality* may be delivered to the client rather than the requested quality level of a segment, based on Consumer Technology Association's *Common Media Client Data* (CMCD) standard [15]. This standard defines the information that a media player can communicate to (within media object requests) and have processed and considered by CDNs. One

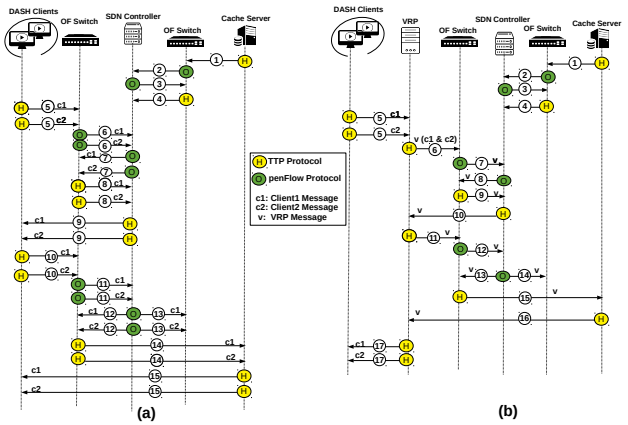

Figure 2: Exchanged messages in (a) SABR and (b) ES-HAS

Table 2: Notation

Input Parameters	
$\mathcal{S}, s$	Set of cache servers and origin server, $s \in \mathcal{S}$
C, c	Set of clients, $c \in C$
$\mathcal{A}, a_q^{c,s}$	Set of available qualities in $\mathcal{S}$, with $a_q^{c,s} = 1$ if quality q requested by client c is available in server s, $a_q^{c,s} = 0$ o/w
$v_q^{c,s}$	$v_q^{c,s} = 1$ if the requested quality q is available in any cache server, $v_q^{c,s} = 0$ o/w
$\mathcal{R}, R_s$	Set of available bandwidth values where R_s is the available bandwidth between VRP and server s
i_c	Quality level requested by client c
m	Integer number to limit the range of potential replacement quality levels for i_c
$\mathcal{K}_c$	Set of eligible quality levels for a quality requested by client c, where $\mathcal{K}_c = \{i_c, i_c + 1, ..., min[i_c + m, q_{max}]\}$
δ_q^c	Size of the segment of quality level q delivered to client c
π_q^c	Bitrate of the quality level $q \in \mathcal{K}_c$
θ	Time slot duration
Variables	
$B_q^{c,s}$	$B_q^{c,s}$=1 if quality q requested by c is served from server s
$T_q^{c,s}$	Required time to fetch quality q for client c from server s
F_c	Deviation of quality level to serve client c w.r.t. i_c
Q_c	Selected quality bitrate to serve client c

piece of information is the "requested maximum throughput that the client considers sufficient for delivery of a content asset". In our system, we interpret this as conveying the information that a client will accept potential replacement (but only better) quality levels in lieu of the actually requested quality of a segment, up to a certain limit set by the client. For the sake of simplification, we assume that each DASH client can request just one segment during each time slot. However, it is possible to consider multiple requests from each client in each time slot without fundamental changes in the proposed model. Note that we will discuss how to determine the time slot duration in Section 5.2. Let set $\mathcal{A}$ denote the cache map (*i.e.*, availability of segments/bitrates) that a VRP receives from the SDN controller, where $a_q^{c,s} = 1$ means that the quality level q requested by client $c \in C$ is available on the server $s \in \mathcal{S}$ (see Table 2 for notations). Moreover, let a VRP's database host set $\mathcal{R}$ containing available bandwidth values between the cache servers and the VRP. In the SOM module, we introduce a mixed-integer linear programming model (MILP) which tries to minimize the segment fetch time if the requested segment quality is available in at least one cache server; otherwise, for each non-cached segment quality, it determines whether to serve the client's request by a replacement quality from a cache server or by the original requested quality from the origin server. We note that AI-based approaches like reinforcement learning could be employed here; however, for the sake of simplicity, a MILP model is proposed, and other approaches will be considered in our future work. The proposed MILP model finds the optimal solution by minimizing the fetch times and the segments' quality level deviations while maximizing the selected quality bitrates. We derive the following constraints that must be satisfied to achieve an optimal solution. Let us define i_c as the quality level requested by client $c \in C$. We also define $\mathcal{K}_c = \{i_c, i_c + 1, ..., min[i_c + m, q_{max}^c]\}$ as the set of eligible (potentially, replacement) quality levels for the segment requested by client c, where m and q_{max}^c denote the maximum deviation from i_c and the maximum quality level of the segment requested by c, respectively. In each optimization interval, we select only one server to serve client c. For this purpose, we introduce binary variable $B_q^{c,s}$, where $B_q^{c,s} = 1$ indicates that quality level q must be served to client c by cache server s. As mentioned earlier, if i_c is available in any cache server, we should force the model to select one cache server to serve the client's request in the original quality by setting the following constraint:

$$\sum_{s \in \{\mathcal{S} - \{Origin\}\}} B_q^{c,s} \times a_q^{c,s} = v_q^{c,s}, \qquad \forall c \in C, q = i_c \quad (1)$$

where $v_q^{c,s} = 1$ if the requested quality $q = i_c$ is available in any cache server; otherwise $v_q^{c,s} = 0$. In the case of a cache miss, *i.e.*, when i_c is not available in any cache server, the VRP can fetch it from the origin server or use other quality levels available in cache servers. Thus, the following constraint must be satisfied:

$$\sum_{s \in \mathcal{S}} \sum_{q \in \mathcal{K}_c} B_q^{c,s} \times a_q^{c,s} = 1, \qquad \forall c \in C \quad (2)$$

The required time to fetch the quality q for client c from server s, denoted by $T_q^{c,s}$, is determined by the following constraint:

$$\delta_q^c \times B_q^{c,s} \leq T_q^{c,s} \times R_s, \qquad \forall c \in C, q \in \mathcal{K}_c, s \in \mathcal{S} \quad (3)$$

where δ_q^c is the size of the quality q requested by client c and R_s is the available bandwidth between the VRP and cache server s. We also define Q_c as the selected quality bitrate to serve client $c \in C$ according to the following constraint:

$$\sum_{s \in \mathcal{S}} \sum_{q \in \mathcal{K}_c} B_q^{c,s} \times \pi_q^c \geq Q_c \qquad \forall c \in C \quad (4)$$

where π_q^c is the bitrate of the selected quality level q for serving client c. To determine the quality deviation when the requested quality i_c is not available in $\mathcal{K}_c$ and a replacement quality has to be found, we introduce the following constraint:

$$\sum_{s \in \mathcal{S}} \sum_{q \in K_c} |i_c - (B_q^{c,s} \times q)| \leq F_c \qquad \forall c \in C \quad (5)$$

Finally, the proposed model is formulated as follows:

$$\begin{aligned} Minimize \quad & \alpha_1 \sum_{c \in C} \sum_{s \in \mathcal{S}} \sum_{q \in \mathcal{K}_c} \frac{T_q^{c,s}}{T^*} + \sum_{c \in C} (\alpha_2 \frac{F_c}{F^*} - \alpha_3 \frac{Q_c}{Q^*}) \qquad (6) \\ s.t. \quad & \text{Constraints} \quad (1) - (5) \\ vars. \quad & T_q^{c,s}, F_c, Q_c \geq 0, B_q^{c,s} \in \{0, 1\} \end{aligned}$$

where T^*, F^*, and Q^* are the maximum values for the fetch time, the quality level deviation, and the quality bitrate for client c, respectively. The SOM runs the above model for all clients' requested qualities in a time-slotted manner to minimize the fetch times and quality level deviations as well as to maximize the qualities delivered to the clients. Moreover, in the objective function (6), we set priorities for $T_q^{c,s}$, F_c, and Q_c by adjusting the weights α_1, α_2, and α_3, respectively. The values of the weights are tuned empirically by considering the network conditions or application policies.

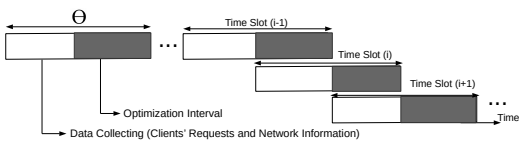

Figure 3: Proposed time slot structure

5 Performance Evaluation

In this section, we evaluate the performance of ES-HAS compared to SABR [12] and pure client-based approaches.

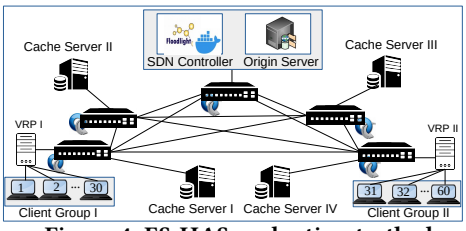

Figure 4: ES-HAS evaluation testbed

5.1 Evaluation Setup

Our testbed consists of 72 nodes running Ubuntu 18.04 LTS inside Xen virtual machines. The proposed network topology is built in the CloudLab [27] environment. As shown in Fig. 4, it includes five OpenFlow (OF) switches, 60 AStream DASH players [3, 21], two VRP servers with the modules described in Section 4. Moreover, we employ four cache servers and one extra server that jointly hosts an origin server and a dockerized SDN controller. The bandwidth values in different paths between each VRP and the cache servers are set to 100, 80, 60, and 40 Mbps, respectively, which explicitly gives higher priority to download segments from the local cache servers. Cache servers I and IV are local cache servers for client groups I and II, respectively, with 100 Mbps bandwidth. The bandwidth values to the origin server are set to 20 Mbps for both VRP servers. Apache [4] and MongoDB [1] with supporting RESTful APIs for cache map exchange are installed on all cache servers. Moreover, Least Recently Used (LRU) is considered in all cache servers as the cache replacement policy. The policy on a cache miss is that the requested quality will be fetched from the origin server only to the related local cache server, *i.e.*, cache server I for client group I and cache server IV for client group II. Floodlight [2] is used as an SDN controller. Among other tasks, it monitors the network to find paths' available bandwidth (in one-second intervals) and as a result assigns paths with the highest available bandwidth between a VRP server and each cache server. For the sake of simplicity, we assume that all clients already joined the network. Ten test videos [23] with 300 seconds durations are used in our experiments. These videos comprise two-second segments in five representations (89k, 0.262M, 0.791M, 2.4M, 4.2Mbps). 60% of the videos' segments are stored in each cache server randomly. All clients run simultaneously in all scenarios. Each client requests one video where $video_1$ is streamed to clients (1,11,21,31,41,51), $video_2$ is streamed to clients (2,12,22,32,42,52), and so on. The time slot duration is set to 52 milliseconds in all experiments. (We will discuss how to obtain the time slot duration in the next subsection.) Two different ABR algorithms, BOLA [29] and SQUAD [31], representing buffer-based and hybrid approaches, are used in all experiments. Python and the PuLP library [5] are employed to implement and solve the proposed MILP model.

5.2 Evaluation of the ES-HAS MILP Model

One of the critical parameters for analyzing the performance of ES-HAS is the time slot duration. In the adaptation process, the client's ABR algorithm estimates the network's bandwidth by measuring the time between sending the request to download a segment and having received the segment's last packet. An additional delay in the request-response interval (due to an overly long time slot duration in the VRP) may convey an incorrect network situation to the client; consequently, the client may request a lower quality level

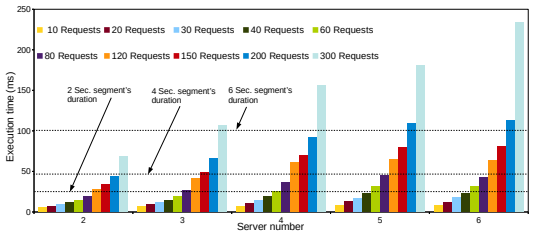

Figure 5: ES-HAS MILP model execution time for different numbers of segment requests and cache servers

for the next segment. Thus, in the first experiment, we investigate various time slot durations for various segment durations in order to find suitable values of θ. For each segment duration of 2, 4, and 6 seconds, we start the experiment with the initial value of $\theta = 10$ ms and gradually increase it in each run. The results show that for time slot durations longer than 52, 95, and 200 ms for 2-, 4-, and 6-sec. segments, respectively, clients start to request lower quality levels. We conclude that these values are the maximum time slot durations without any negative effect on the clients' adaptation behavior. In the second experiment, we numerically investigate the scalability of the proposed MILP model execution. As illustrated in Fig. 3, each time slot includes data collecting and optimization intervals. In the worst case, requests arriving at the beginning of the optimization interval should wait to be processed in the next optimization interval. In other words, a request could possibly wait for two optimization intervals plus a collection interval. To avoid optimization interval overlapping, in this study, we assume the optimization interval should be less than or equal to $\frac{\theta}{2}$. Therefore, based on the first experiment, this value should be less than 26, 47, and 100 milliseconds for 2-, 4-, and 6-sec. segments, respectively. We measure the MILP execution time for various numbers of clients' requests and cache servers (Fig. 5). Considering the optimization interval duration, the proposed MILP model can handle about 60, 100, and 210 different requests for 2-, 4-, and 6-sec. segments, respectively, in a topology with four cache servers (see Fig. 5). The number of requests decreases when increasing the number of cache servers. We now calculate the maximum number of clients that each VRP can handle in a topology with four cache servers. As we discussed earlier, in our experiments, each client sends 150 requests to download 150 segments with 2 seconds duration. We showed that the maximum time slot duration should be less than or equal to 52 ms; that means we have about 5700 ($\frac{300}{0.052}$) time slots in each experiment. Assuming a uniform distribution, each client sends a request in a given time slot with a probability of 0.026 ($\frac{150}{5700}$). For instance, in the case of 60 requests as the maximum number of requests that can be handled by the proposed MILP model in each time slot, one VRP can serve up to a maximum of 2300 clients and send 60 distinct requests per time slot ($2300 \leq \frac{60}{0.026}$).

5.3 Evaluation of the ES-HAS Framework

To investigate the behavior of the ES-HAS framework regarding different values of α_1, α_2, and α_3, we define three metrics: *ACS*: the usage percentage of cache servers with the shortest fetch time (since the first term of the objective function (Eq. 6) forces the model to select a cache server with the shortest fetch time); *AMD*: the average (for different m) of the maximum deviation between

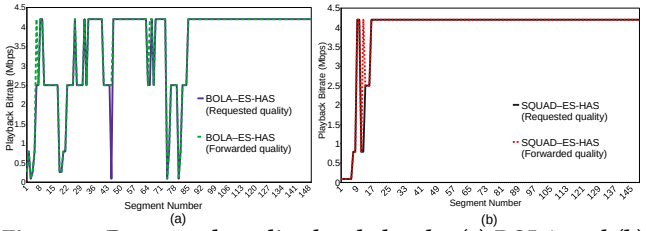

Figure 6: Requested quality levels by the (a) BOLA and (b) SQUAD algorithms vs. forwarded quality levels for client 1

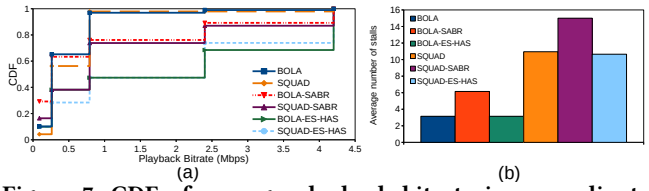

Figure 7: CDF of average playback bitrate in pure client-based ABR, SABR, and ES-HAS (a), and average number of stalls (b) for 60 clients

requested quality and forwarded quality (second term of the objective function (Eq. 6); and *AQB*: the average of the video quality bitrate for all received segments in Mbps (third term of the objective function (Eq. 6). In the first scenario, we investigate the reactions of clients' ABR algorithms when their decisions are overwritten by replacement qualities. For this purpose, we set α_1, α_2, α_3, and m to 0.1, 0.1, 0.8, and 3, respectively, to force the model to select appropriate replacement qualities in cache miss situations. All requested qualities and forwarded qualities are logged by the VRP. As shown in Fig. 6 (client I), the requested quality levels by the BOLA and SQUAD algorithms are overwritten multiple times by better replacement qualities (*e.g.*, segments 46 and 15 requested by BOLA (Fig. 6(a)) and SQUAD (Fig. 6(b)) respectively); however, that does not have a negative impact on the overall playback quality of subsequent segments. Other clients have a similar reaction to the replacement quality. We extend our experiments by various values of α and m and investigate their impact on the aforementioned metrics. Table 3 shows the results for group I of clients. As expected, most of the requested quality levels are delivered from cache server I (denoted c_1) for a high α_1 value, which forces the system to emphasize low fetch time ($\alpha_1 = 0.8$, $\alpha_2 = 0.1$, $\alpha_3 = 0.1$). On the other hand, setting $\alpha_2 = 0.8$ forces the model to deliver the quality levels as requested by the clients and keeps AMD to zero ($\alpha_1 = 0.1$, $\alpha_2 = 0.8$, $\alpha_3 = 0.1$). By setting α_3 to a high value, the model can serve higher quality levels to clients ($\alpha_1 = 0.1$, $\alpha_2 = 0.1$, $\alpha_3 = 0.8$). Although increasing m and α_3 increases the AMD, the model performs better in AQB since it has a wider range to select replacement quality levels ($\alpha_1 = 0.1$, $\alpha_2 = 0.1$, $\alpha_3 = 0.8$, and $m = 3$).

In the second scenario, we compare the performance of ES-HAS with the pure client-based (video players in client groups I and II directly send their requests to CDN edge servers I and IV, respectively) and the SABR [12] approaches. Because [12] used different components than we have in our testbed, we implemented the SABR approach and reproduced its results without using its available source code. The α_1, α_2, α_3, and m values are set to 1, 0, 0, and 0, respectively, to have fair comparisons among the schemes; *i.e.*, the VRP only transmits the original requested quality from

Table 3: Impact of m, α_1, α_2, and α_3 on MILP model behavior

m	Metric	(α_1,α_2,α_3): (.8,.1,.1)	(α_1,α_2,α_3): (.1,.8,.1)	(α_1,α_2,α_3): (.1,.1,.8)
1	ACS	c_1 : 56%, c_2 : 25%	c_1 : 51%, c_2 : 31%	c_1 : 46%, c_2 : 36%
	AMD	0	0	0
	AQB	3.73	3.73	3.73
2	ACS	c_1 : 55%, c_2 : 26%	c_1 : 53%, c_2 : 29%	c_1 : 54%, c_2 : 29%
	AMD	1	0	1
	AQB	3.75	3.73	3.75
3	ACS	c_1 : 56%, c_2 : 36%	c_1 : 52%, c_2 : 31%	c_1 : 52%, c_2 : 32%
	AMD	1	0	.437
	AQB	3.75	3.75	3.8

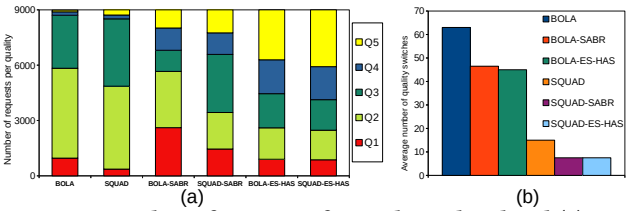

Figure 8: Number of requests for each quality level (a), and average number of quality switches (b) for 60 clients

the optimal cache server. As illustrated in Fig. 7 and Fig. 8(b), the ES-HAS framework outperforms the pure client-based method in terms of playback bitrate and the number of quality switches since it fetches segments from the cache server with the shortest fetch time. Although SABR and ES-HAS show (almost) identical results in terms of the number of quality switches for both ABR methods (Fig. 8(b)), ES-HAS results in better performance in terms of playback bitrate (Fig. 7(a)), the number of requests for the highest quality level (Fig. 8(a)), and the number of stalls compared to SABR (Fig. 7(b)). Recognizing similar requests (video/segment/quality) and sending only one request instead of several requests to the selected cache server, plus employing an optimization-based approach for the cache/segment selection policy are the main reasons for ES-HAS performance improvements over SABR.

6 Conclusion

This paper leverages the SDN and NFV paradigms to propose the ES-HAS framework providing network assistance for HTTP adaptive video streaming. We introduce VNF components named VRPs at the edge of the network. ES-HAS (via the SDN controller) provides VRPs with information on network conditions and available video sources (cache servers). In addition, VRPs collect clients' requests for video segments and aggregate them in time slots. Using this information, VRPs run an optimization model in a time-slotted manner to help clients get their requested segment quality levels from cache servers with the shortest fetch times or receive from cache servers better replacement quality levels with minimum deviation from the original requested quality levels. We implement the proposed framework and its modules on a cloud-based large-scale testbed consisting of 60 clients. We conduct experiments in different scenarios, evaluate the MILP model's behavior, and compare the results with another state-of-the-art approach. Experimental results demonstrate that, on average, ES-HAS outperforms SABR in terms of playback bitrate and the number of stalls by at least 70% and 40%, respectively. Extending our proposed framework for edge caching and redesigning the proposed MILP model to improve more QoE parameters and establish users' fairness are possible future work directions.

Acknowledgments

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

References

[1] 2007. MongoDB. Retrieved 2021-03-07 from https://www.mongodb.com
[2] 2012. Floodlight. Retrieved 2021-03-07 from https://github.com/floodlight/floodlight
[3] 2013. AStream: A rate adaptation model for DASH. Retrieved 2021-03-07 from https://github.com/pari685/AStream
[4] 2020. Apache HTTP Server. Retrieved 2021-03-07 from https://httpd.apache.org
[5] 2020. PuLP. Retrieved 2021-04-13 from https://pypi.org/project/PuLP/
[6] Saamer Akhshabi, Lakshmi Anantakrishnan, Ali C Begen, and Constantine Dovrolis. 2012. What happens when HTTP adaptive streaming players compete for bandwidth?. In *Proceedings of the 22nd International Workshop on Network and Operating System Support for Digital Audio and Video*. 9–14.
[7] Alcardo Alex Barakabitze, Arslan Ahmad, Rashid Mijumbi, and Andrew Hines. 2020. 5G network slicing using SDN and NFV: A survey of taxonomy, architectures and future challenges. *Computer Networks* 167 (2020), 106984.
[8] Alcardo Alex Barakabitze, Nabajeet Barman, Arslan Ahmad, Saman Zadtootaghaj, Lingfen Sun, Maria G Martini, and Luigi Atzori. 2019. QoE management of multimedia streaming services in future networks: a tutorial and survey. *IEEE Communications Surveys & Tutorials* (2019).
[9] Abdelhak Bentaleb, Ali C Begen, and Roger Zimmermann. 2016. SDNDASH: Improving QoE of HTTP adaptive streaming using software defined networking. In *Proceedings of the 24th ACM International Conference on Multimedia*. 1296–1305.
[10] Abdelhak Bentaleb, Ali C Begen, Roger Zimmermann, and Saad Harous. 2017. SDNHAS: An SDN-enabled architecture to optimize QoE in HTTP adaptive streaming. *IEEE Transactions on Multimedia* 19, 10 (2017), 2136–2151.
[11] Abdelhak Bentaleb, Bayan Taani, Ali C Begen, Christian Timmerer, and Roger Zimmermann. 2018. A survey on bitrate adaptation schemes for streaming media over HTTP. *IEEE Communications Surveys & Tutorials* 21, 1 (2018), 562–585.
[12] Divyashri Bhat, Amr Rizk, Michael Zink, and Ralf Steinmetz. 2017. Network assisted content distribution for adaptive bitrate video streaming. In *Proceedings of the 8th ACM on Multimedia Systems Conference*. 62–75.
[13] Margaret Chiosi, Don Clarke, Peter Willis, Andy Reid, James Feger, Michael Bugenhagen, Waqar Khan, Michael Fargano, Chunfeng Cui, Hui Deng, et al. 2012. Network functions virtualisation: An introduction, benefits, enablers, challenges and call for action. In *SDN and OpenFlow World Congress*, Vol. 48. SN, 202.
[14] Cisco. February 2019. Cisco Visual Networking Index: Forecast and Trends, 2017–2022. *White Paper* (February 2019).
[15] Consumer Technology Association. 2020. *CTA Specification: Web Application Video Ecosystem - Common Media Client Data. CTA-5004.* Technical Report. https://cdn.cta.tech/cta/media/media/resources/standards/pdfs/cta-5004-final.pdf
[16] Oussama El Marai, Tarik Taleb, Mohamed Menacer, and Mouloud Koudil. 2017. On improving video streaming efficiency, fairness, stability, and convergence time through client–server cooperation. *IEEE Transactions on Broadcasting* 64, 1 (2017), 11–25.
[17] Alireza Erfanian, Farzad Tashtarian, Reza Farahani, Christian Timmerer, and Hermann Hellwagner. 2020. On Optimizing Resource Utilization in AVC-based Real-time Video Streaming. In *2020 6th IEEE Conference on Network Softwarization (NetSoft)*. IEEE, 301–309.
[18] Alireza Erfanian, Farzad Tashtarian, and Mohammad H. Yaghmaee. 2018. On Maximizing QoE in AVC-Based HTTP Adaptive Streaming: An SDN Approach. In *2018 IEEE/ACM 26th International Symposium on Quality of Service (IWQoS)*. 1–10. https://doi.org/10.1109/IWQoS.2018.8624161
[19] Te-Yuan Huang, Nikhil Handigol, Brandon Heller, Nick McKeown, and Ramesh Johari. 2012. Confused, timid, and unstable: picking a video streaming rate is hard. In *Proceedings of the 2012 Internet Measurement conference*. 225–238.
[20] Junchen Jiang, Vyas Sekar, and Hui Zhang. 2012. Improving fairness, efficiency, and stability in HTTP-based adaptive video streaming with festive. In *Proceedings of the 8th International Conference on Emerging Networking Experiments and Technologies*. 97–108.
[21] Parikshit Juluri, Venkatesh Tamarapalli, and Deep Medhi. 2015. SARA: Segment aware rate adaptation algorithm for dynamic adaptive streaming over HTTP. In *2015 IEEE International Conference on Communication Workshop (ICCW)*. IEEE, 1765–1770.
[22] Diego Kreutz, Fernando MV Ramos, Paulo Esteves Verissimo, Christian Esteve Rothenberg, Siamak Azodolmolky, and Steve Uhlig. 2014. Software-defined networking: A comprehensive survey. *Proc. IEEE* 103, 1 (2014), 14–76.
[23] Stefan Lederer, Christopher Müller, and Christian Timmerer. 2012. Dynamic adaptive streaming over HTTP dataset. In *Proceedings of the 3rd Multimedia Systems Conference*. 89–94.
[24] Hwanwook Lee, Yunmin Go, and Hwangjun Song. 2018. SDN-assisted HTTP adaptive streaming over Wi-Fi network. In *2018 Fifth International Conference on Software Defined Systems (SDS)*. IEEE, 205–210.
[25] Zhi Li, Xiaoqing Zhu, Joshua Gahm, Rong Pan, Hao Hu, Ali C Begen, and David Oran. 2014. Probe and adapt: Rate adaptation for HTTP video streaming at scale. *IEEE Journal on Selected Areas in Communications* 32, 4 (2014), 719–733.
[26] Konstantin Miller, Emanuele Quacchio, Gianluca Gennari, and Adam Wolisz. 2012. Adaptation algorithm for adaptive streaming over HTTP. In *2012 19th International Packet Video Workshop (PV)*. IEEE, 173–178.
[27] Robert Ricci, Eric Eide, and CloudLab Team. 2014. Introducing CloudLab: Scientific infrastructure for advancing cloud architectures and applications. *; login:: the magazine of USENIX & SAGE* 39, 6 (2014), 36–38.
[28] Iraj Sodagar. 2011. The MPEG-DASH standard for multimedia streaming over the internet. *IEEE Multimedia* 18, 4 (2011), 62–67.
[29] Kevin Spiteri, Rahul Urgaonkar, and Ramesh K Sitaraman. 2016. BOLA: Near-optimal bitrate adaptation for online videos. In *IEEE INFOCOM 2016-The 35th Annual IEEE International Conference on Computer Communications*. IEEE, 1–9.
[30] Emmanuel Thomas, MO van Deventer, Thomas Stockhammer, Ali C Begen, M-L Champel, and Ozgur Oyman. 2016. Applications and deployments of server and network assisted DASH (SAND). (2016).
[31] Cong Wang, Amr Rizk, and Michael Zink. 2016. SQUAD: A spectrum-based quality adaptation for dynamic adaptive streaming over HTTP. In *Proceedings of the 7th International Conference on Multimedia Systems*. 1–12.