electronics

MDPI

Article

SAND/3: SDN-Assisted Novel QoE Control Method for Dynamic Adaptive Streaming over HTTP/3

Luis Guillen [1,*], Satoru Izumi [1], Toru Abe [1,2] and Takuo Suganuma [1,2]

[1] Graduate School of Information Sciences, Tohoku University, Miyagi-Sendai 980-8577, Japan
[2] Cyberscience Center, Tohoku University, Miyagi-Sendai 980-8577, Japan
* Correspondence: lguillen@ci.cc.tohoku.ac.jp; Tel.: +81-22-217-5080

Received: 3 July 2019; Accepted: 1 August 2019; Published: 4 August 2019

Abstract: Dynamic Adaptive Streaming over HTTP (DASH) is a widely used standard for video content delivery. Video traffic, most of which is generated from mobile devices, is shortly to become the most significant part of Internet traffic. Current DASH solutions only consider either client- or server-side optimization, leaving other components in DASH (e.g., at the transport layer) to default solutions that cause a performance bottleneck. In that regard, although it is assumed that HTTP must be necessarily transported on top of TCP, with the latest introduction of HTTP/3, it is time to re-evaluate its effects on DASH. The most substantial change in HTTP/3 is having Quick UDP Internet Connections (QUIC) as its primary underlying transport protocol. However, little is still know about the effects on standard DASH client-based adaption algorithms when exposed to the future HTTP/3. In this paper, we present SAND/3, an SDN (Software Defined Networking)-based Quality of Experience (QoE) control method for DASH over HTTP/3. Since the official deployment of HTTP/3 has not been released yet, we used the current implementation of Google QUIC. Preliminary results show that, by applying SAND/3, which combines information from different layers orchestrated by SDN to select the best QoE, we can obtain steadier media throughput, reduce the number of quality shifts in at least 40%, increase the amount downloaded content at least 20%, and minimize video interruptions compared to the current implementations regardless of the client adaption algorithm.

Keywords: SDN; DASH; QoE; HTTP/3; QUIC

1. Introduction

Hypertext Transport Protocol (HTTP) version 3 or HTTP/3 is the name given by IETF to HTTP over Quick UDP Internet Connections (QUIC) [1]. HTTP/3 subsumes the benefits of HTTP/2 with TLS, reduced connection delay (0-RTT) on top of UDP, and at the same time offers TCP features (e.g., packet retransmission and congestion control). Being HTTP a fundamental component in Dynamic Adaptive Streaming over HTTP (DASH) [2], it is essential to analyze the effects on its overall performance for future internet traffic, which, according to the Visual Networking Index [3], is supposed to be mostly comprised by video streams from mobile devices.

It is worth noting that, although it is widely assumed that DASH must be transported on top of TCP, there are no specific restrictions on using other protocols. For example, Google reported a significant improvement in YouTube and Search services when deployed on top of QUIC [1]. However, little is still known about the impact of this protocol over other DASH schemes. While the authors of [4–7] showed a positive influence, others, such as Bhat et al. [8], conversely showed a detriment in the user perceived quality, also known as Quality of Experience (QoE).

The broad idea of DASH is that servers store different versions of video content, all of which are listed in an XML file called Media Presentation Description (MPD). Based on the network status and other parameters, clients can request a representation (version) of the video file. However,

the DASH standard does not mandate a specific way to perform the adaption. Consequently, there are several approaches on how to adapt the video quality request, from hybrid quality adaption [9] to comprehensive optimization models [10]. Nonetheless, a single-parameter-based adaption (i.e., client or server) offers poor QoE. While a purely client-based adaption might not reach optimal performance due to its restricted knowledge, a purely server-side adaption would disregard users' features. Moreover, due to the ever-changing network conditions, frequent adaption requests based on inaccurate information lead to instability.

In that regard, using a centralized control plane would improve the video delivery [11,12]. Therefore, a Software Defined Networking (SDN)-based solution will adequately help to improve the adaption by providing a means of communication that includes an overview of the entire infrastructure. SDN allows the segregation of the control plane from the forwarding plane; therefore, due to its centralized nature and its global knowledge of the network, it can act as an intermediary between the DASH clients and servers, improving the adaption schemes. Therefore, the main goal of this paper is to evaluate the effects of SDN combined with HTTP/3 in DASH adaption schemes. To this aim, we implemented a holistic solution that considers all components involved in a DASH system to improve the overall QoE. The primary hypothesis to test was that, by applying the proposed approach of using HTTP/3 assisted by SDN and taking into account cross-layer collaboration, the overall QoE in DASH would significantly improve compared to current solutions that use single-component optimization on top of TCP.

The specific contribution of this paper is two-folded:

- In Section 3, we describe the overall design and interaction of the proposed QoE Control method for DASH over HTTP/3, whose main strength is its simplicity, and how SDN can orchestrate network-, transport-, and user-level components.
- In Section 5, we present a comprehensive evaluation of some of the most common adaption algorithms and the effects on QoE when they use HTTP/3 compared to the current deployment.

Based on preliminary results in Section 6, the proposed approach presents various advantages for DASH, i.e., higher and steadier average bitrate, minimizing the number of video stalls (freezes), and reducing the number of video quality shifts compared to traditional methods.

2. Related Work

2.1. TCP-Based SDN Solutions for DASH

Although there is a large body of knowledge about DASH and how to perform the adaption [11,13]. in this section, we briefly present the related work focused on SDN-based solutions to improve video delivery in DASH over traditional TCP. Some of the early work [14–16] proposed improving video quality by dynamically routing prioritized video stream flows over OpenFlow (OF) networks. Although using a dynamic priority route segregates the traffic, relying on network-only variables will be significantly constrained. In that regard, the authors of [17–19] improved the overall QoE by integrating *client feedback*, which constitutes an additional layer of information to perform the adaption. Therefore, by adding information from higher-layers into the adaption process will provide more flexible support to improve the user's QoE. The authors of [20,21] went even further and proposed an SDN-based architecture for DASH; both solutions presented innovative advantages for quality management. However, their deployment increased the network overhead. SDNHAS [22], an improved version of the method in [21], presents an elegant solution that combines multi-layer optimization based on user categories. However, they all are still constrained by simply relying on HTTP over TCP at the transport layer.

2.2. Non-TCP-Based SDN Solutions for DASH

Relying HTTP upon single TCP connections brings along known problems (e.g., increased delay and head-of-line blocking). Therefore, methods that use variations at the transport level show an

improvement in the overall QoE in DASH. For instance, the authors of [23,24] used Multipath TCP (MPTCP) [25], an extension of TCP, to transmit *subflows* of DASH streams over multiple paths. The results were significantly better than using single TCP connections. However, despite the benefits of using MPTCP, both end-points need kernel modification. Furthermore, there are other conditions to fulfill, which will over-complicate the deployment in DASH or even have no effect, as reported by James et al. [26]. Few other authors experimented with QUIC and DASH assisted by SDN. Hussein et al. [7] proposed a QUIC-aware SDN architecture to handle congestion control, load balancing, and security limitations of QUIC. Although it was one of the first to test the integration of these technologies, it is not clear how the proposed architecture help DASH systems, apart from the initial connection. Hayes et al. [5] also marginally used SDN with MPTCP and QUIC to reduce the initial connection and an adaptive protocol selection based on performance feedback. Although it is an interesting scheme, the constant exchangeability between protocols would create a service disruption in highly active services, which will lead to detrimental QoE. As observed, related work present innovative and practical solutions to improve video QoE in DASH using SDN; however, there are still some open issues summarized as follows:

- *(P1) Single-component optimization*: Most solutions focus on a single component optimization (network, client, or server).
- *(P2) Transport ossification*: Most solutions assume TCP as the only protocol for end-to-end communication for HTTP, which creates a performance bottleneck.

Our work complements and extends the existing work by holistically tackling these issues. Specifically, as opposed to the methods in Section 2.1, by using HTTP/3 as the transport protocol, the proposed method does not suffer from the inherent problems of using HTTP over TCP, i.e., the head-of-line (HOL) blocking issue. Moreover, complementing the efforts of the methods in Section 2.2, our proposed method includes user-level information to perform a more fine-tuned adaption.

3. SDN-Assisted Novel DASH QoE Control Method over HTTP/3 (SAND/3)

To solve the problems stated above and offer users a better QoE, in this section, we describe our SDN-Assisted Novel QoE control method for DASH over HTTP/3 (SAND/3). The overall architecture of the proposed scheme is depicted in Figure 1. As observed, SAND/3 combines user-, device-, service- and network-level information, handled by three modules whose function is as follows:

- *User module*: This module collects the end-user identity and the associated devices from a third-party service in which the user is subscribed (e.g., Netflix) and stores them in a repository called *user profile*. We assume that this information will be available and provided by the third-party service. The profile consists of a registry of tuples comprised of a user identification number, associated devices, and preferences. Thus, when a device requests a video segment throughout an HTTP message, the *device manager* fetches the associated profile and collects the basic specifications (e.g., display size, available memory, buffer size, type of device, and subscription plan) and assigns a user priority for the current video playback. For instance, a user with a premium subscription to Netflix streaming a video on a SmartTV would have a higher priority than a user with a free account on a mobile device since their quality requirements are different.
- *Network module*: This module performs traffic engineering, by routing the packets via the most suitable paths that offer the best Quality of Service (QoS). To do so, first, the network status is continuously monitored by the *Network Monitor* sub-module, which collects the statistics of all the elements in the network, allowing a better estimation of the available resources. As part of this module, the *Topology Manager* updates the network device status, and it is triggered every time there is a change in the topology. Finally, the *Routing Handler* calculates the end-to-end paths to handle the traffic according to the priority assigned to each user. For simplicity, we create k

different paths using a modified version of the well-known k-maximum disjoint paths Suurballe's algorithm [27], where k is the number of categories in the application policy, e.g., if the quality categories are high, normal, and low, then $k = 3$. The cost of the path (weight) is calculated using Dijkstra's shortest path algorithm, having each link a cost based on network parameters or a combination of them, i.e., available bandwidth, delay, packet loss and so forth. In the current implementation, we use the delay as the only factor to calculate the path cost, but these can be easily extended to more elaborate weights.

- *Application module*: Based on the user profile, the current state of the network, and the specific service policies, the QoE Manager sub-module recommends the most suitable settings for the transmission, which is handled by the Transport Handler sub-module. Note that the Transport Handler sub-module is in charge of performing the transport connection using QUIC protocol.

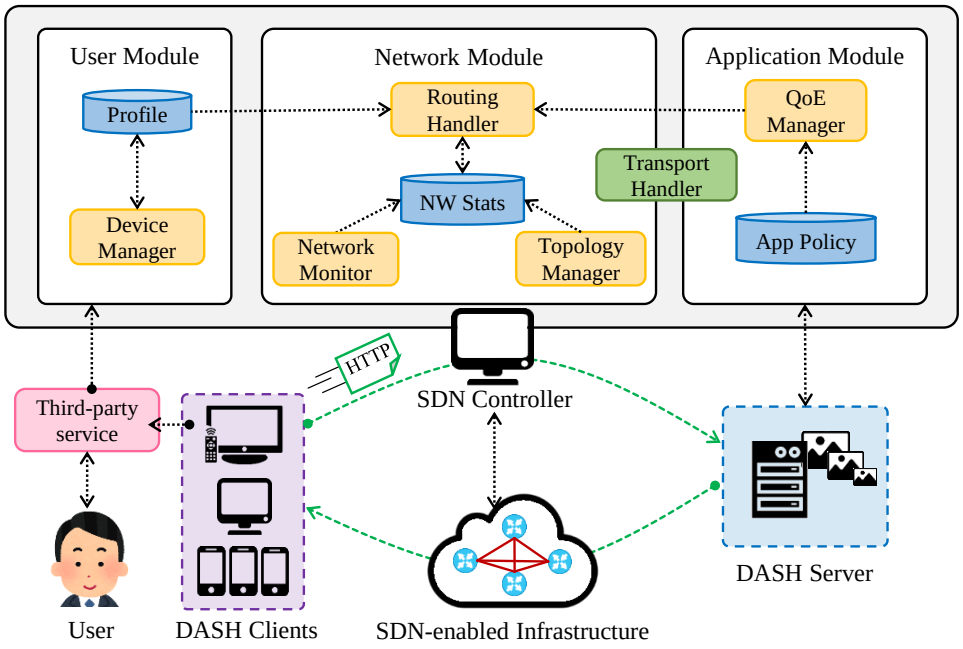


Figure 1. SAND/3 architecture.

4. An Overview of SAND/3

The network model is as follows, we consider a directed graph $G = (V, E)$, where V and E are the set of network devices and links connecting them, respectively. Each element in $v_i \in V$ is linked by an edge $e_{i,j} \in E$ between v_i and v_j. Moreover, $b_{i,j}$, $d_{i,j}$, and $c_{i,j}$ represent the bandwidth, delay, and cost, respectively of $e_{i,j}$. For simplicity, $c_{i,j}$ is calculated as the $b_{i,j} \times d_{i,j}$. If s and t are the source and destination devices, respectively, a path $p_{s,t} \in P$ is the shortest path from a source v_s to the destination v_t such that the cost $C_{s,t}$, the sum of link costs, is minimum. Finally, given a number of categories $k \in 1, 2, 3, ...$, the set P_k is comprised by the first k paths $p_{s,t}$.

The flowchart depicted in Figure 2 shows the overall process, which broadly works as follows: Firstly, when an HTTP request is received from the DASH client, it is captured at the edge switch and sent to the controller. Then, the system fetches the associated *User Profile* from the local repository (*Profile*) based on the information of HTTP headers—host, destination, user-agent, alt-svc, etc. In the current implementation, we assume single homogeneous devices for all users, and, therefore, we use the source and destination host as the identifier. However, more fine-grained information can be extracted from headers to match the request to a specific user properly. It is also important to note that, at this step, based on the information in the headers, the *Transport Handler* can determine whether both the client and server support HTTP/3 or use regular TCP connections. Then, the systems read the service policies from the local repository called *App Policy*, among those policies it is necessary to

obtain the available user categories, which is then used for calculating paths with different priorities according to the user type. The mapping process from a user HTTP request to a path is shown in the dotted region in Figure 2. One of the most essential processes is the calculation of the user category, which can be done using various approaches, from advanced Machine Learning (ML) or other Artificial Intelligence (AI) as done by Bentaleb et al. [22], to other optimization heuristics as done by Herguner et al. [23]. However, for simplicity, in the current implementation, we attach this value as a parameter on the *User Profile*, leaving a more elaborate selection process as future work. Then, to create the paths, we calculate k (number of categories) paths so that the request is matched to the ith path, where i is the number of the category to which the user belongs. In case there is no available path, the traffic will be sent via the default shortest path. Finally, once the path has been selected and written in the adequate switches, the DASH client can start the transmission, and request/send them over the same connection until the playback is finished (see right-hand side of Figure 2). Note that the mapping is done once per transmission. However, it will be updated in the case there is a significant change in the topology (e.g., network device failure and link disconnection).

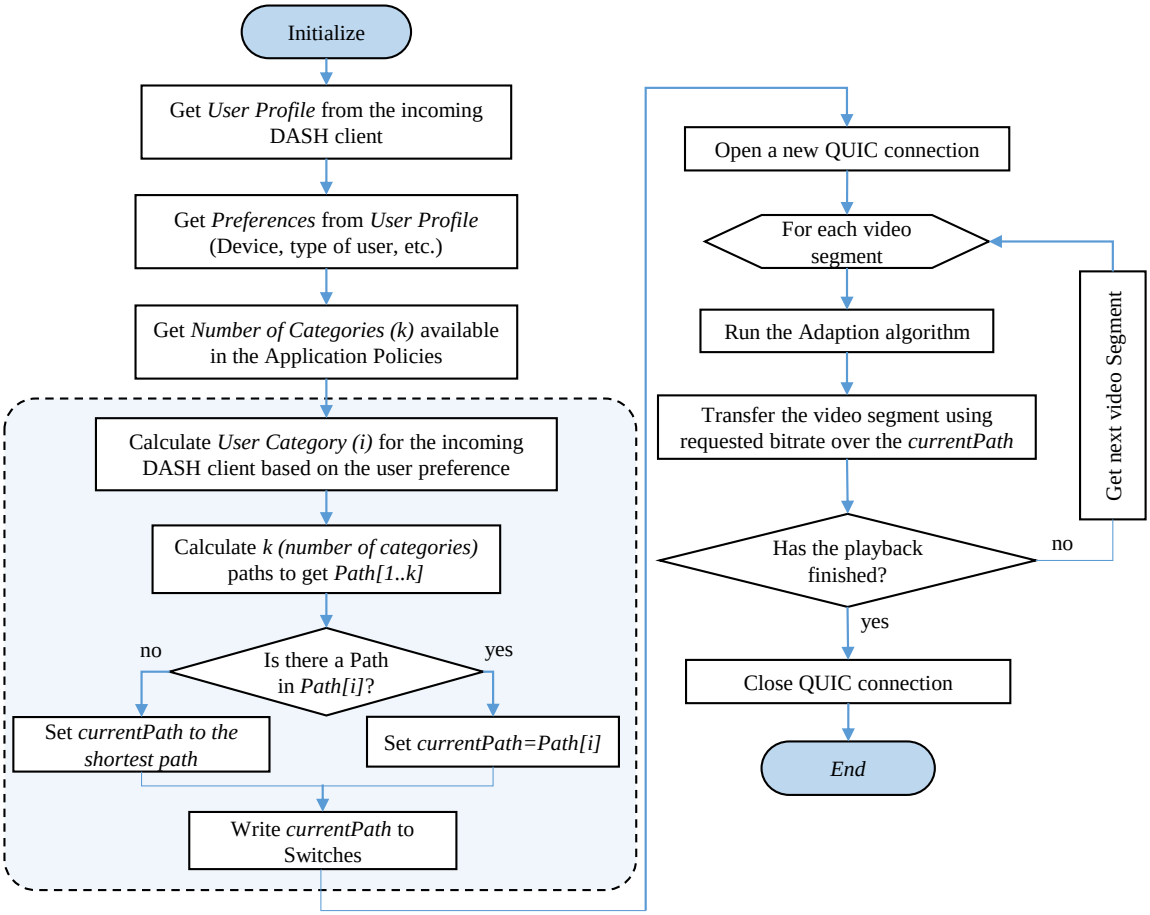


Figure 2. Overall flowchart process for QoE mapping to network segment from a source DASH client.

5. Evaluation

5.1. Test Environment

We used an emulated environment for both DASH and network, hosted on a single virtual machine running Ubuntu 16.04 LTS with 16 Gb of memory and six cores Intel Xeon(R) E5-2650 v4 of 2.20 GHz. The network was emulated using Mininet [28] v.2.2.2 which runs OpenFlow v.1.3 (OF) with the values described in Table 1. For simplicity, we used a 3 × 3 grid topology, as shown in

Figure 3. In the case of HTTP/3, since the official deployment was not available yet at the moment of the writing of this paper [29], we used the test QUIC client/server provided by Google [30]. The QUIC server runs the latest version (43 at the moment of the implementation), which, although it differs in some aspects to the IETF implementation, can serve as a preview on how the overall interaction of HTTP over QUIC would work on DASH. In the case of the DASH clients, we used a modified version of AStream [31], which is a Python-based emulated video player, with the QUIC client integrated as a sub-module as done by Arisu et al. [6]. Moreover, we modified the official QUIC server by automatically adding the required header tags (i.e., alternative protocol, port number, URL) per file in the server directory, which is mandatory in the test QUIC server. To compare with the current approach (using HTTP over TCP only), we used the built-in HTTP Server module from Python at the server side, and *libcurl* TCP client integrated into AStream. In addition, note that we kept the same connection for the entire transmission—as QUIC would do within the same connection—rather than opening a new connection per request. Finally, as the SDN controller, we used OpenDaylight (ODL) Beryllium SR2 and implemented the functions of each module as an internal application.

Table 1. Experimental parameters.

Parameter	Value
Topology	3 × 3 Grid
Bandwidth ($b_{i,j}$)	10 Mbps for all links
Delay ($d_{i,j}$)	Randomly assigned (1, 3, and 5 ms)
# of DASH Clients	3 (Xterm) emulating PCs
DASH Server	Simple HTTP Server
Controller	ODL Beryllium SR1

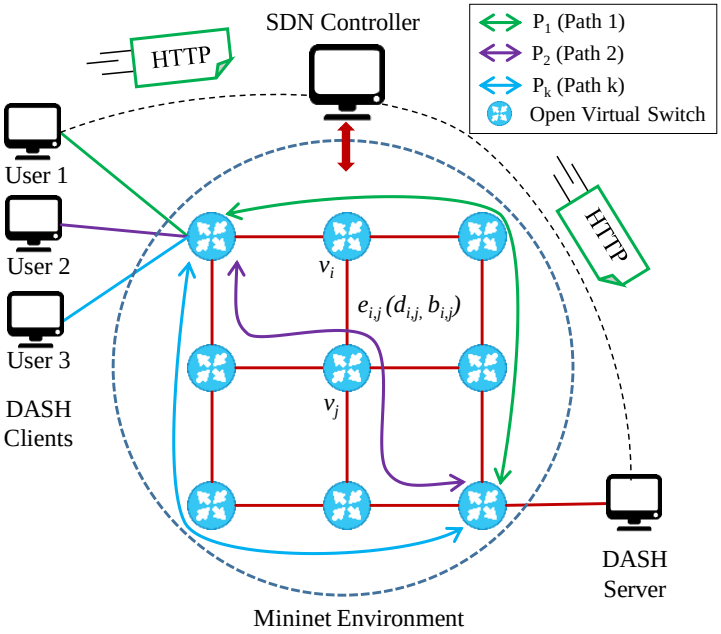


Figure 3. Testbed setup.

5.2. Use Case

Since streaming solutions over HTTP were originally developed for Video-on-Demand (VoD) [13], we used VoD as the use-case; however, we believe that the proposed solution can also be applicable for live streaming. To conduct the experiments, we used Big Buck Bunny (BBB) from a publicly available DASH dataset [32] by Lederer et al. [33]. In this dataset, the full-length BBB video (of approximately 600 s) is encoded in twenty different representations and segment (video chunks) sizes from 1 to

15 s ranging from bitrates of 420 × 360 pixels at 45 Kbps to 1920 × 1080 pixels at 3.9 Mbps. For the experiments, we used only a 4-s segment size, as it is an intermediate value between encoding and flexibility in the bitrate adaptation. To test the approach, we compared the performance of SAND/3 by using the following adaption bitrate (ABR) algorithms in the DASH clients:

- *Throughput-based adaption (TBA)* [34]: TBA starts the transmission by requesting the lowest available bitrate and, based on the average network throughput, the following segments will be selected in an additive increasing, multiplicative decreasing (AIMD) manner.
- *Buffer-Based adaption (BBA)* [35]: While other bitrate adaption approaches focus on throughput capacity prediction, BBA uses only the buffer occupancy to request the initial segment and then, if needed, estimates the capacity. BBA decreases the re-buffering events while the playback bitrate increases. The second version of this algorithm (BBA-2) was part of a large-scale experiment on Netflix, which achieved about 10–20% decreased re-buffering. Specifically for the experiments, after some preliminary results, instead of using a buffer size equal to the video segment size, as would usually be done, we used three times that value to make a fair comparison with the other compared algorithms. In Table 2, we describe a complete list of the BBA parameters used in the experiment.
- *Segment Aware Rate Adaptation (SARA)* [36]: Contrary to other ABR algorithms, which assume equal segment sizes, SARA takes into account the variation in segment and buffer sizes, so that it allows a more accurate prediction of the next segment. The configuration parameters used in the experiment are described in Table 3.

Table 2. List of BBA parameters used in the experiment.

Parameter	Value
Buffer size (B_{max})	12
Initial buffer	2
Initial factor	0.75
Reservoir (r)	0.2
Cushion (cu)	0.75

Table 3. List of SARA parameters used in the experiment.

Parameter	Value
Sample count	5
Initial buffering occupancy (B_{curr})	5
Re-buffering bound I	1
Buffer lower threshold (B_{α})	5
Buffer upper threshold (B_{β})	10

Each client was assigned a profile and a single device with the same specifications regarding the memory and CPU, emulated by Xterm terminals in Mininet. The priority level was different for each user; for simplicity, we numbered the priorities in ascending order so that high-, medium-, and low-priority would map to 0, 1, and 2, respectively. Note that, in the current implementation, these values were set statically in the user profile; a more comprehensive solution would require, for instance, a classification based on all the parameters, as done by Bentaleb et al. [22]. The experiment was conducted as follows. For each client ABR algorithm (namely TBA, SARA, and BBA), we played BBB on all the devices randomly starting within an approximate 30 s interval. In each run, all clients used the same adaption algorithm using the three variants:

- TCP-only approach: In this approach, the transmission of HTTP is over a TCP stream, as the current default solution would behave.
- QUIC-only approach: HTTP is sent on top of QUIC, emulating the way HTTP/3 would work but only changing the transport protocol without further improvements.

- SAND/3: The proposed scheme was used on top of QUIC and applying the process explained in previous sections.

In the first two cases, we left the default connectivity settings, which includes the end-to-end routing using simply the shortest path route. Note that, due to some performance limitations in the QUIC test server, not suitable for a large number of users [30], we only used three clients; nevertheless, since the primary objective of the experiment was to analyze the behavior and the feasibility of the proposal rather than a large-scale test, the current setup would suffice as a proof-of-concept.

5.3. QoE Metrics

To objectively measure the user QoE, there are five factors that influence the most in a DASH system: the number times the video freezes (stalls), the number of video quality shifts, media throughput (bitrate), start-up delay, and the fairness at shared bottlenecks [13,37]. However, since the start-up delay and fairness are mostly related to live-streaming, they are perceived less critical than the other variables [38]. Therefore, in this paper, we focus on the number of stalls, media throughput and the number of video quality shifts to measure the QoE.

6. Results

6.1. Number of Stalls

Concerning the number of stalls, which occurs when the video stops due to a buffer underrun, Table 4 shows the results obtained per adaption algorithm and their combined duration. As observed, there were stalls only in the TCP-only (current default) approach—the other approaches (QUIC and our approach) did not present any stalls during the playback whatsoever—which is a favorable result since a stall is the most detrimental factor for QoE [11]. In addition, note that most of the stalls occurred on users from the lower priority category. However, in the case of SARA, even users from the first categories suffered from stalls, and although there were few occurrences the combined time was considerable (16–26 s), which shows how unfair bandwidth competition can affect all users as the number of parallel requests increases.

Table 4. Number of stalls per user category using the TCP-only approach.

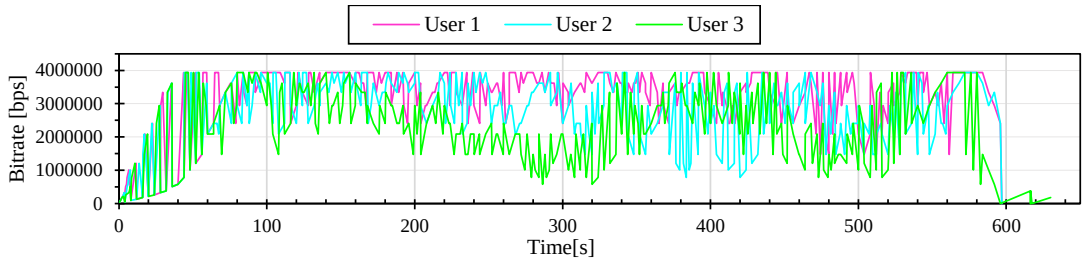
User Category	Adaption Algorithm					
	TBA		SARA		BBA	
	# of Stalls	Time [s]	# of Stalls	Time [s]	# of Stalls	Time [s]
1	0	0	2	21.1	0	0
2	0	0	1	16	0	0
3	1	20	4	26	1	2.46

6.2. Media Throughput

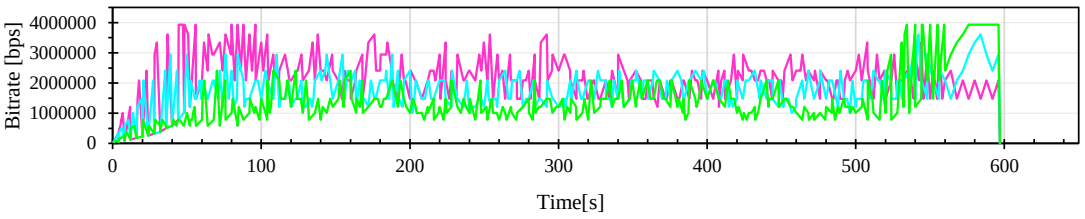
Concerning the media throughput (received video bitrate), depending on the adaption algorithm and the approach used, the results were relatively different. Figures 4–6 we show the media throughput obtained for each type of user using TBA, BBA, and SARA respectively.

Initially, in Figure 4a, we can see that, although TBA increased the requested bitrate progressively when using TCP-only approach, once higher quality segments were achieved and as new users started to request better quality representations, the video quality decreased significantly. This considerable quality decrease was due to the shared bandwidth bottleneck at the edge of the topology, and the unfair bandwidth competition. On the other hand, users that used QUIC-only, as shown in Figure 4b, received relatively low-quality segments for most of the transmission; however, at no time did the clients suffer from buffer underruns. Moreover, the bandwidth distribution was somehow evenly distributed among clients. Finally, in Figure 4c, for the case of our approach, the bitrate changes were

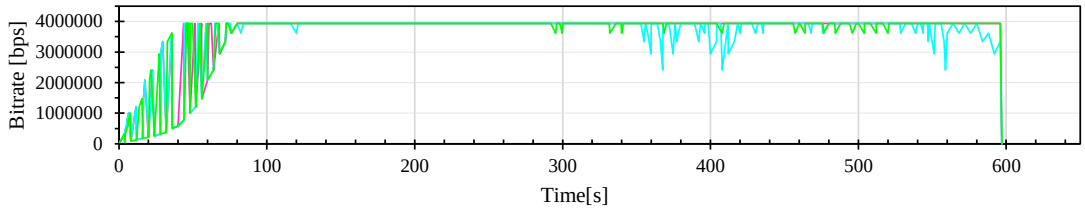
quickly improved due to the characteristics of TBA. Moreover, once the clients converged to the best available video quality, the playback was steady and stayed, for the most part, in the highest video quality even for users with lower priority, which shows the benefits of the proposed approach for all user levels.



(**a**) Received bitrate using TCP-only



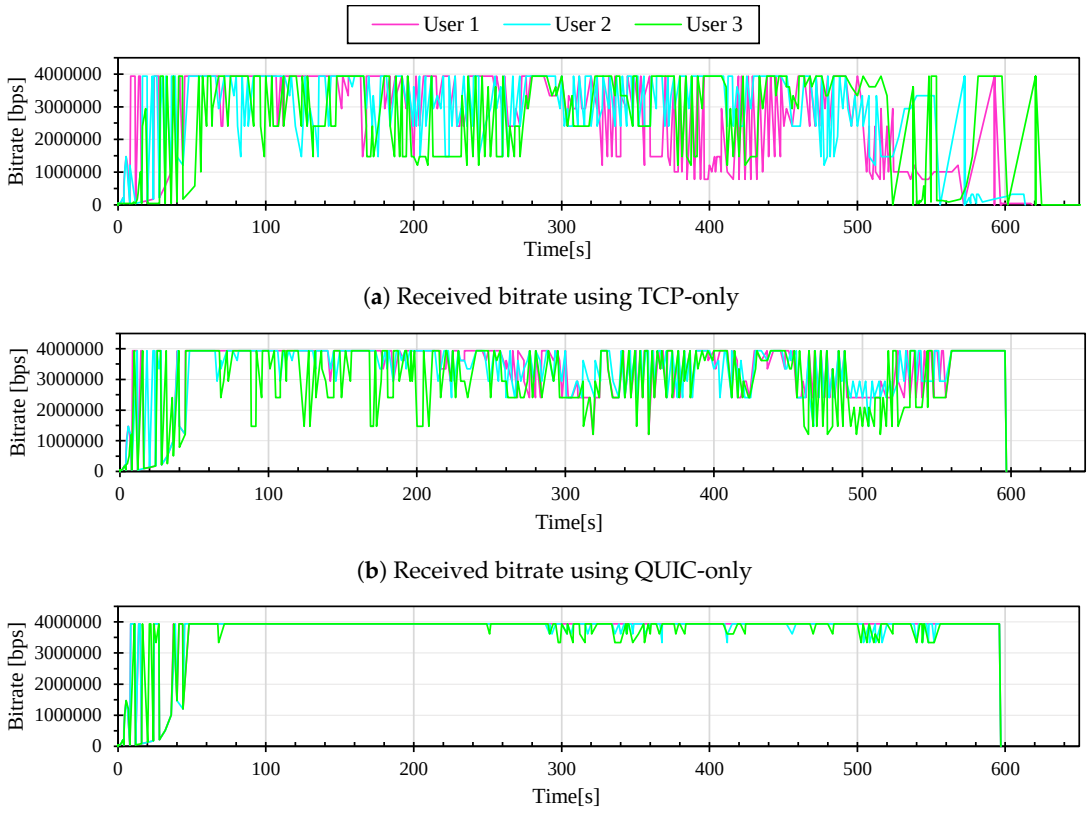
(**b**) Received bitrate using QUIC-only



(**c**) Received bitrate using the proposed approach

Figure 4. Media throughput obtained for three users at 10 Mpbs links using TBA adaption algorithm.

When SARA was used as the DASH adaption algorithm, as shown in Figure 5, since the algorithm considers the variable sizes of the different video representations, the convergence time was faster than the case of TBA. However, due to the congestion in the shared lines used in the TCP-, and QUIC-only approaches, the estimation was done using non-accurate information, and therefore the requested bitrate could not be handled by either of the approaches, leading to a highly unstable playback, as can be observed in Figure 5a,b. Due to the conservative nature of SARA, especially in terms of the *Delayed Download* mechanism, the video quality is gradually increased and maintained in the same one for a certain amount of time to avoid unnecessary downloads. However, by the time the segment is to be transmitted, the network conditions might have already changed so that the link will be overloaded downloading a bigger file until the threshold reaches the lower bound. On the other hand, by using our approach, once the optimal bitrate is selected, the feedback can estimate the available resources more accurately since the network status is continuously monitored on all network devices, which led to a steady and smooth playback with the highest video quality for all users, as shown in Figure 5c.



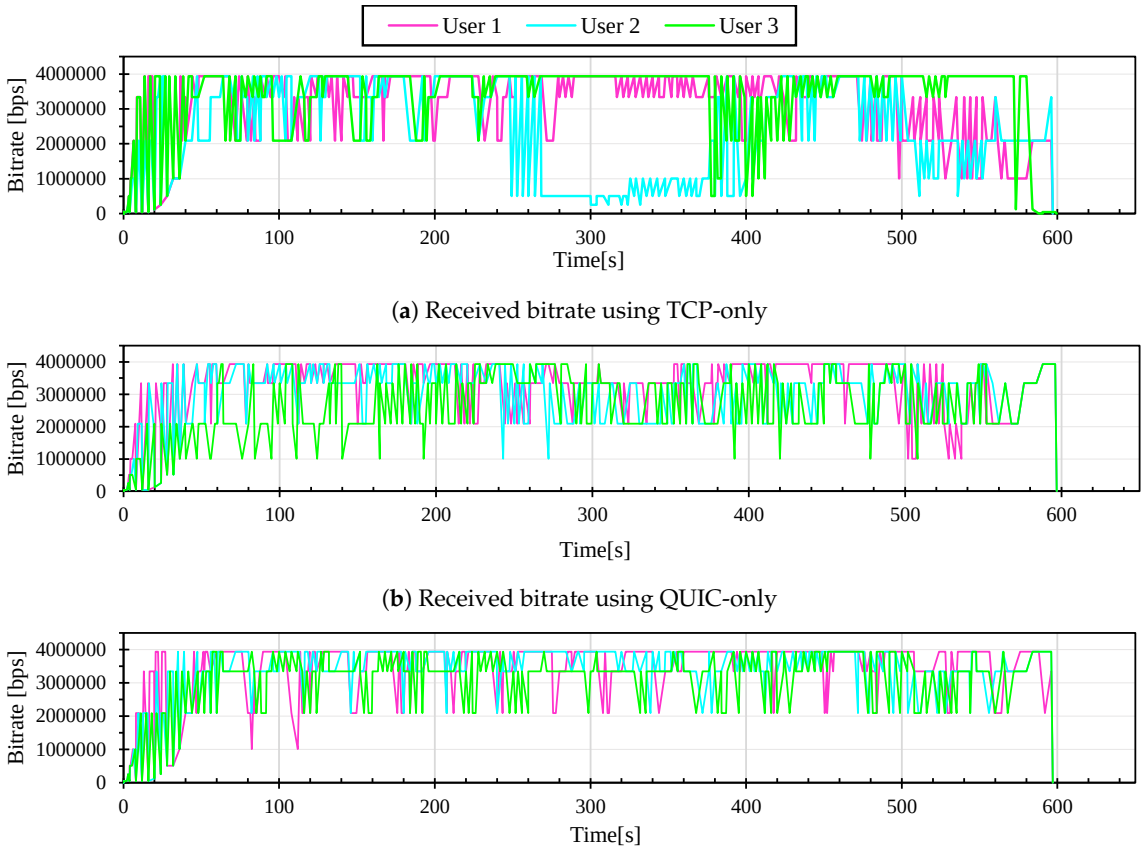
(**a**) Received bitrate using TCP-only

(**b**) Received bitrate using QUIC-only

(**c**) Received bitrate using the proposed approach

Figure 5. Media throughput obtained for three users at 10 Mpbs links using SARA adaption algorithm.

Finally, in Figure 6, we show the received bitrate when using BBA as the DASH adaption algorithm. Note that the bitrate variability is higher than the other algorithms. BBA has a different approach, which focuses on the current buffer occupancy rather than throughput estimation. Thus, it tends to request segments with higher bitrates for a longer time. However, when the streams compete with other requests, the congestion level increases considerably, leading to overloaded links and consequently higher loss-rate. As observed in Figure 6a, which depicts the received bitrate using TCP, there was a point in the playback (approximately at Second 250) where the congestion level was so high that the bitrate suffers from a phenomenon called *ON–OFF period*. In this state, the requested video quality oscillates between the highest and lowest, creating instability among the other DASH clients due to an inaccurate buffer estimation. In the case of QUIC, although there was some interference between Users 2 and 3, BBA was capable of recovering relatively quickly, as observed in Figure 6b. Lastly, in the case of our approach, as shown in Figure 6c, the bitrate was between the upper level of cushion (cu) and the maximum buffer capacity (B_{max}) for the most of the transmission, which allowed a smooth video playback without any buffer underruns.



(**a**) Received bitrate using TCP-only

(**b**) Received bitrate using QUIC-only

(**c**) Received bitrate using the proposed approach

Figure 6. Media throughput obtained for three users at 10 Mpbs links using BBA adaption algorithm.

6.3. Video Quality Shifts

Concerning the number of video quality shifts, which refers to the change of video representation, users perceive dissatisfaction when the video abruptly changes to a considerably lower bitrate; therefore, a progressive increase of quality and minimizing the downshifts is desirable to have a good QoE. Figure 7 show the number of both up- and down-shifts events in every approach. As observed in Figure 7a–c, regardless on the adaption algorithm used (i.e., TBA, SARA, or BBA respectively), our approach was able to considerably minimize the downshift events at least for the user with the highest priority (User 1). However, note that, in the case of the number of the downshifts using BBA (Figure 7c), for Users 2 and 3, the number is higher using our approach than when using TCP. Nonetheless, as previously shown in Figure 6, the difference in the decrease of video quality was much bigger when using TCP than in our approach.

Regarding the number of upshifts, even though this variable is tightly coupled with the downshift events, the strategy used in each algorithm will significantly influence on how fast the bitrate will converge again to the best possible quality before a downshift event. Most of the adaption algorithms will opt for an aggressive decrease and progressive increase, having, therefore, more upshift events than downshifts, as shown in Figure 7d–f, which depict the number of upshift events using TBA, SARA, and BBA respectively. The average improvement achieved was around 45%, and 60% for the downshifts, and 40% and 50% for the upshift events compared to the TCP-only, and QUIC-only approaches, respectively, which is a significant improvement to the overall QoE.

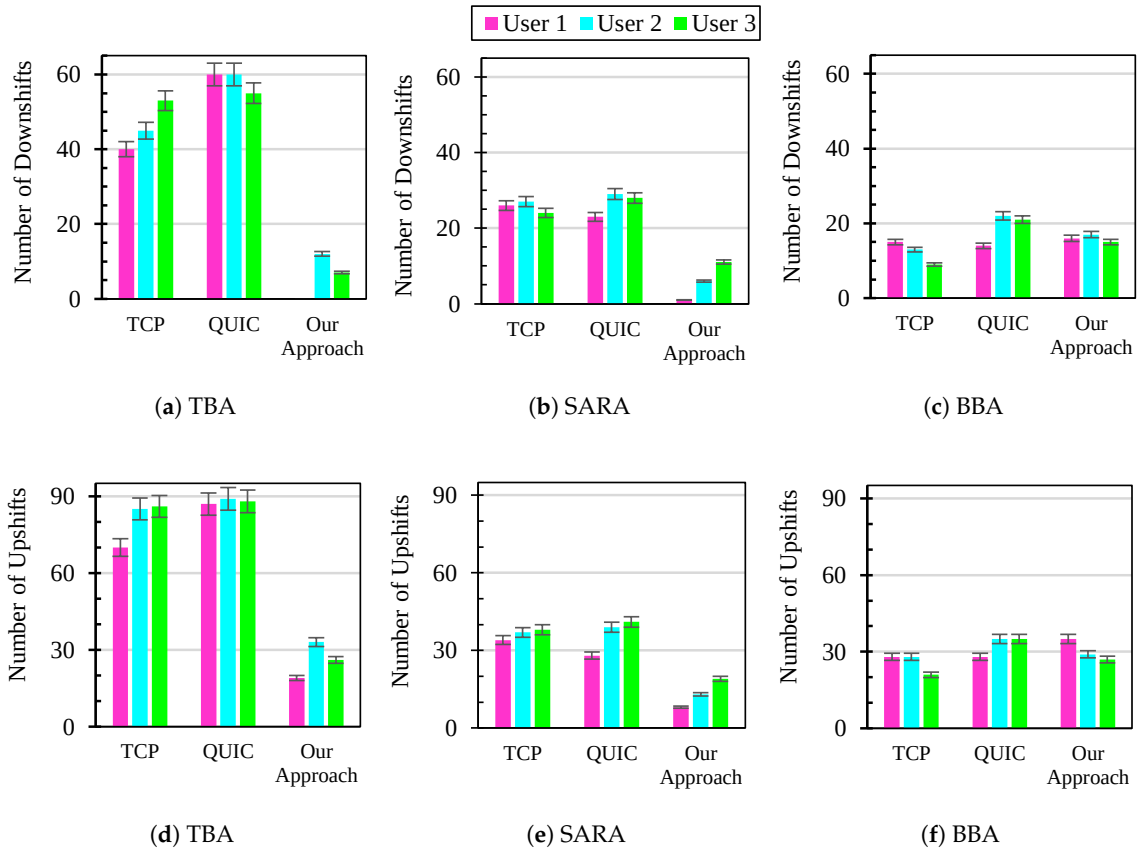
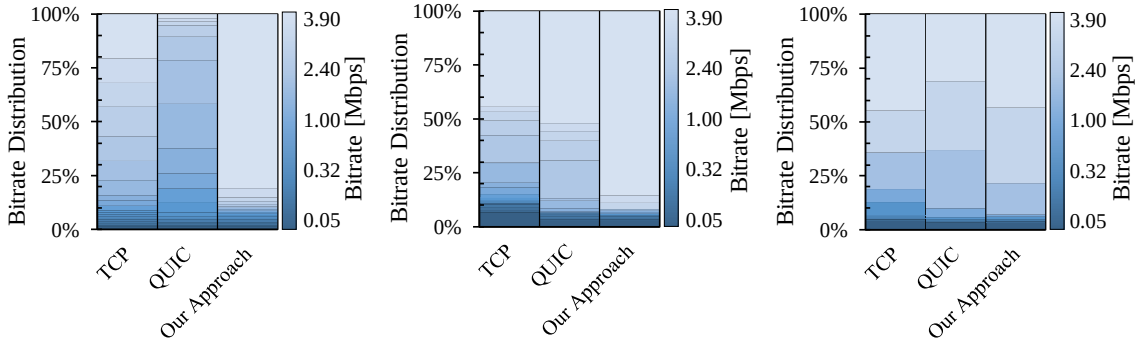


Figure 7. Number of up- and down-shifts per approach using different adaption algorithms.

6.4. Average Downloaded Video Files

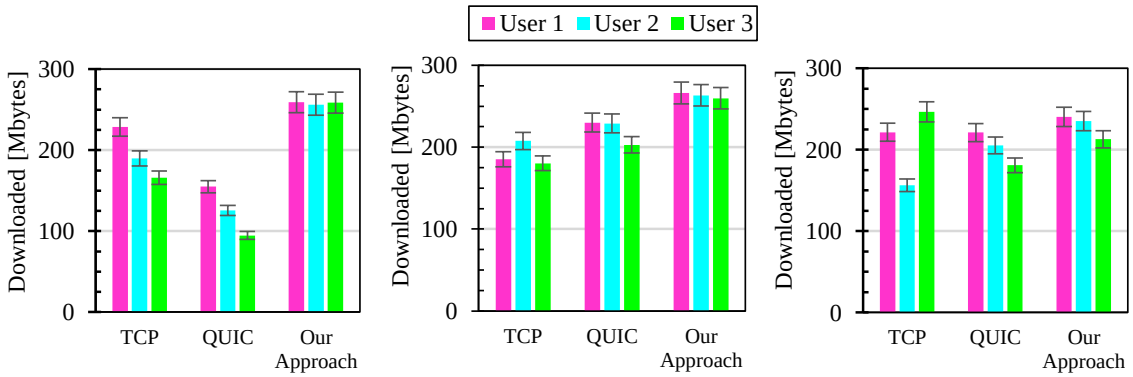
Figure 8 depicts the heat-map of the average bitrate distribution of each of the approaches using the different adaption algorithms throughout the entire video. As observed, regardless of the algorithm, the bitrate distribution is considerably wider towards the highest video quality segments (light blue) when using the proposed approach. In fact, when using TBA and SARA, about 85% of the downloaded segments belonged to that category (Figure 8a,b), while, in the case of BBA (Figure 8c), it was about 45%, which was marginally better than the other approaches but still the second best video representation covered a larger part of the downloaded segments. This shows that it is possible to guarantee the best quality regardless of the user category or the adaption algorithm used. Since a better representation is stored in a bigger file, the amount of downloaded content was also significantly higher than when using TCP- or QUIC-only approaches, as shown in Figure 9a,b, respectively. For the most part, the distribution among the clients was rather even; however, note that there is a slight progressive decrease in the case of BBA (Figure 9c) as each user was affected by the number of clients and the increased congestion. Nevertheless, the approach achieved an average of 20% improvement compared to the TCP-only approach, and 25% compared to the QUIC-only approach.

These results confirm that the proposed approach provides a better overall QoE for all users in the different categories, while effectively using the available resources.



(**a**) TBA (**b**) SARA (**c**) BBA

Figure 8. Heat map of average downloaded video representation per approach.



(**a**) TBA (**b**) SARA (**c**) BBA

Figure 9. Downloaded content per approach.

7. Discussion

Based on the obtained results, we could observe that using our approach, or even only implementing HTTP/3, will greatly benefit DASH performance. Even though in the current paper we used a test server from Google's QUIC implementation, the benefits can be extrapolated when the official IETF implementation becomes available. As observed, the main advantage would be the avoidance of stalling events regardless of the adaption algorithm used by the DASH client, even if they share a bottleneck at the edge of the topology or the network resources are scarce. This is, of course, an inherent benefit of using QUIC protocol at the transport level, which was precisely designed to improve the performance on users with constraint performance in terms of congestion, and loss percentage [1]. Changing just the transport protocol by itself will not be enough, as some other factors will also have a detrimental effect on the QoE, e.g., taking into account user categories or sudden changes in the network. However, although the inclusion of additional control levels will overload the network devices, there have been some recent proposals—e.g., *Segment Routing* (SR) [39]—that can help alleviate the flow-table exponential increase. The overall idea of SR is to divide the end-to-end path into multiple parts called "segments", which will be updated based on pre-established policies, and it is a useful technique to relief the flow-table overload in SDN implementations, as shown in [24].

An additional inherent advantage, especially for mobile users, would be solving the mobility issue. The control and identification of the transmission in QUIC is done by using a *Connection ID*, making therefore possible to migrate a transmission from an end-point that had a change in lower-layers (UDP, IP) and continue their process, even the in case of a vertical (i.e., interface) or horizontal (i.e., network) handover [29], as can be observed in [6] where the authors explored this phenomenon.

It is also important to mention that, although we did not modify the default values of TCP or UDP for the experiments (e.g., congestion window), we believe that, by adjusting this parameter, as suggested by Kakhki et al. [40], the results can be further improved for the approaches that use QUIC.

It is worth noting that using HTTP over QUIC has some drawbacks in the current implementations. For instance, Google and Facebook stated that a wide-scale deployment would require about twice the amount of CPU to cope with the traffic load compared to current HTTP/2 [29]. Of course, this performance issue might be reduced by further optimizing the hardware and software (i.e., when the changes are applied in the kernel) to the UDP stack in a rather short time, since this area is not as explored as in the case of TCP.

Another point to consider is that, since a centralized-control approach—which combines information otherwise segregated (i.e., user, network, service, and application)—might be seen challenging to implement, there are already authors who presented feasible solutions. For instance, Liotou et al. [41] proposed a solution that allows effective communication between Video Service Providers (VSPs) and mobile network operators' (MNOs), which enables feedback towards a better network-aware video segment selection.

Finally, note that we only focused on a fully SDN-based infrastructure; the current Internet is made of different intertwined technologies, and therefore the interaction between those non-SDN domains needs further study.

8. Conclusions and Future Work

In this paper, we present SAND/3, an SDN-Assisted QoE control method for DASH over HTTP3, which comprises a multi-layer collaborative optimization at the user, application, transport, and networking levels. The proposed approach combines the best state-of-the-art technologies to support quality adaption and improve DASH performance on top of HTTP/3. We implemented the proposed architecture over an SDN-based infrastructure using ODL as the controller in a Linux environment. Moreover, we evaluated its feasibility using an emulated DASH client using different adaption bitrate algorithms. Preliminary results show that, by considering end-user categories to manage the video segment's traffic over QUIC, the overall QoE improves not only regarding the media throughput but also reducing the number of stalls, and the number of abrupt downshifts of video quality compared to current TCP-only and QUIC-only approaches. However, although the QUIC-only approach showed no significant improvements for DASH by itself, we can conclude that QUIC (and HTTP/3) features might help to improve DASH performance but it needs further support.

Future directions this work might go are as follows: First, we still need to test the impact of the approach in different scenarios, e.g., using other adaption algorithms or heterogeneous clients. Then, we plan to deploy the system using larger network environments, and test using real implementations of DASH systems and HTTP/3. Finally, it might also be interesting to look at the impact of recent approaches, such as Multipath QUIC [42], as it could combine the benefits of QUIC streams multiplexing over multiple paths and their impact on future Internet applications, in particular video streaming.

Author Contributions: Conceptualization, T.S. and L.G.; investigation, data curation, methodology, software and writing—original draft preparation, L.G.; and writing—review and editing, and supervision, S.I., T.A. and T.S.

Conflicts of Interest: The authors declare no conflict of interest.

References

1. Langley, A.; Riddoch, A.; Wilk, A.; Vicente, A.; Krasic, C.; Zhang, D.; Yang, F.; Kouranov, F.; Swett, I.; Iyengar, J.; et al. The QUIC transport protocol: Design and internet-scale deployment. In Proceedings of the Conference of the ACM Special Interest Group on Data Communication (SIGCOMM'17), Los Angeles, CA, USA, 21–25 August 2017; pp. 183–196.

2. *ISO/IEC 23009-1:2014, Information Technology—Dynamic Adaptive Streaming over HTTP (DASH)—Part 1: Media Presentation Description and Segment Formats*; International Organization for Standardization: Geneva, Switzerland, 2014.
3. CISCO. *The Zettabyte Era: Trends and Analysis*; White Paper; CISCO, 2017. Available online: https://files.ifi.uzh.ch/hilty/t/Literature_by_RQs/RQ%20102/2015_Cisco_Zettabyte_Era.pdf (accessed on 3 July 2019)
4. Timmerer, C.; Bertoni, A. Advanced transport options for the dynamic adaptive streaming over HTTP. *arXiv* **2016**, arxiv:1606.00264.
5. Hayes, B.; Chang, Y.; Riley, G. Omnidirectional Adaptive Bitrate Media Delivery using MPTCP/QUIC over an SDN Architecture. In Proceedings of the IEEE Global Communications Conference (GLOBECOM2017), Singapore, 4–8 December 2017; pp. 1–6.
6. Arisu, S.; Begen, A.C. Quickly Starting Media Streams Using QUIC. In Proceedings of the 23rd Packet Video Workshop, Amsterdam, The Netherlands, 12–15 June 2018; pp. 1–6.
7. Hussein, A.; Kayssi, A.; Elhajj, I.H.; Chehab, A. SDN for QUIC: An Enhanced Architecture with Improved Connection Establishment. In Proceedings of 33rd Annual ACM Symposium on Applied Computing, Pau, France, 9–13 April 2018; pp. 2136–2139.
8. Bhat, D.; Rizk, A.; Zink, M. Not so QUIC: A Performance Study of DASH over QUIC. In Proceedings of the 27th Workshop on Network and Operating Systems Support for Digital Audio and Video, Taipei, Taiwan, 20–23 June 2017; pp. 13–18.
9. Mok, R.K.P.; Luo, X.; Chan, E.W.W.; Chan, R.K.C. QDASH: A QoE-aware DASH System. In Proceedings of the 3rd Multimedia Systems Conference, Chapel Hill, NC, USA, 22–24 February 2012; pp. 11–22.
10. Tashtarian, F.; Erfanian, A.; Varastehy, A. S2VC: An SDN-based Framework for Maximizing QoE in SVC-Based HTTP Adaptive Streaming. *Comput. Netw.* **2018**, *146*, 33–46. [CrossRef]
11. Seufert, M.; Egger, S.; Slanina, M.; Zinner, T.; Hoßfeld, T.; Tran-Gia, P. A survey on quality of experience of HTTP adaptive streaming. *IEEE Commun. Surv. Tutor.* **2015**, *17*, 469–492. [CrossRef]
12. Ganjam, A.; Jiang, J.; Liu, X.; Sekar, V.; Siddiqi, F.; Stoica, I.; Zhan, J.; Zhang, H. C3: Internet-scale control plane for video quality optimization. In Proceedings of the 12th USENIX Conference on Networked Systems Design and Implementation, Oakland, CA, USA, 4–6 May 2015; pp. 131–144.
13. Petrangeli, S.; van der Hooft, J.; Wauters, T.; De Turck, F. Quality of Experience-centric Management of Adaptive Video Streaming Services: Status and Challenges. *ACM Trans. Multimed. Comput. Commun. Appl.* **2018**, *14*, 31. [CrossRef]
14. Egilmez, H.E.; Civanlar, S.; Tekalp, A.M. An optimization framework for QoS-enabled adaptive video streaming over openflow networks. *IEEE Trans. Multimed.* **2013**, *15*, 710–715. [CrossRef]
15. Laga, S.; Van Cleemput, T.; Van Raemdonck, F.; Vanhoutte, F.; Bouten, N.; Claeys, M.; De Turck, F. Optimizing scalable video delivery through OpenFlow layer-based routing. In Proceedings of the 2014 Network Operations and Management Symposium (NOMS), Krakow, Poland, 4–9 May 2014; pp. 1–4.
16. Owens, H.; Durresi, A. Video over Software-Defined Networking (VSDN). In Proceedings of the 16th International Conference on Network-Based Information Systems, Gwangju, Korea, 4–6 September 2013; pp. 44–51.
17. Jarschel, M.; Wamser, F.; Hohn, T.; Zinner, T.; Tran-Gia, P. SDN-based Application-Aware Networking on the Example of YouTube Video Streaming. In Proceedings of the 2nd European Workshop on Software Defined Networks, Berlin, Germany, 4–6 October 2013; pp. 87–92.
18. Nam, H.; Kim, K.; Kim, J.; Schulzrinne, H. Towards QoE-aware Video Streaming using SDN. In Proceedings of the Communications Software, Services and Multimedia Symposium (Globecom2014), Austin, TX, USA, 8–12 December 2014; pp. 1317–1322.
19. Georgopoulos, P.; Elkhatib, Y.; Broadbent, M.; Mu, M.; Race, N. Towards Network-wide QoE Fairness Using OpenFlow-assisted Adaptive Video Streaming. In Proceedings of the Workshop on Future Human-Centric Multimedia Networking (FhMN13), Hong Kong, China, 16 August 2013; pp. 15–20.
20. Mkwawa, I.; Barakabitze, A.A.; Sun, L. Video Quality Management over the Software Defined Networking. In Proceedings of the IEEE International Symposium on Multimedia, San Jose, CA, USA, 11–13 December 2016; pp. 554–564.

21. Bentaleb, A.; Begen, A.C.; Zimmermann, R. SDNDASH: Improving QoE of HTTP Adaptive Streaming Using Software Defined Networking. In Proceedings of the 24th ACM International Conference on Multimedia, Amsterdam, The Netherlands, 15–19 October 2016; pp. 15–19.
22. Bentaleb, A.; Begen, A.C.; Zimmermann, R.; Harous, S. SDNHAS: An SDN-Enabled Architecture to Optimize QoE in HTTP Adaptive Streaming. *IEEE Trans. Multimed.* **2017**, *19*, 2136–2151. [CrossRef]
23. Herguner, K.; Kalan, R.S.; Cetinkaya, C.; Sayit, M. Towards QoS-aware Routing for DASH Utilizing MPTCP over SDN. In Proceedings of the 2017 IEEE Conference on Network Function Virtualization and Software Defined Networks (NFV-SDN), Berlin, Germany, 6–8 November 2017; pp. 1–7.
24. Barakabitze, A.A.; Mkwawa, I.; Sun, L.; Ifeachor, E. QualitySDN: Improving Video Quality using MPTCP and Segment Routing in SDN/NFV. In Proceedings of the International Conference on Network Softwarization (NetSoft2018), Montreal, QC, Canada, 25–29 June 2018; pp. 182–186.
25. Raiciu, C.; Paasch, C.; Barre, S.; Ford, A.; Honda, M.; Duchene, F.; Bonaventure, O.; Handley, M. How hard can it be? designing and implementing a deployable multipath TCP. In Proceedings of the 9th USENIX Symposium on Networked Systems Design and Implementation, San Jose, CA, USA, 25–27 April 2012; pp. 1–14.
26. James, C.; Halepovic, E.; Wang, M.; Jana, R.; Shankaranarayanan, N. K. Is Multipath TCP (MPTCP) Beneficial for Video Streaming over DASH? In Proceedings of the IEEE 24th International Symposium on Modeling, Analysis and Simulation of Computer and Telecommunication Systems (MASCOTS), London, UK, 19–21 September 2016; pp. 331–336.
27. Suurballe, J.W.; Tarjan, R.E. A quick method for finding shortest pairs of disjoint paths. *Networks* **1984**, *14*, 325–336. [CrossRef]
28. Lantz, B.; Heller, B.; McKeown, N. A Network in a Laptop: Rapid Prototyping for Software-Defined Networks. In Proceedings of the 9th ACM Workshop on Hot Topics in Networks, Monterey, CA, USA, 20–21 October 2010.
29. Stenberg, D. HTTP/3 Explained. Available online: Https://daniel.haxx.se/http3-explained/ (accessed on 3 July 2019).
30. Chromium-Playing with QUIC. Available online: Https://www.chromium.org/quic/playing-with-quic (accessed on 17 November 2018).
31. AStream: A Rate Adaptation Model for DASH. Available online: Https://github.com/pari685/AStream (accessed on 15 December 2018).
32. ITEC DASH Dataset. Available online: Http://www-itec.uni-klu.ac.at/ftp/datasets/DASHDataset2014/ (accessed on 23 January 2019).
33. Lederer, S.; Muller, C.; Timmerer, C. Dynamic adaptive streaming over HTTP dataset. In Proceedings of the 3rd ACM Multimedia Systems Conference, Chapel Hill, NC, USA, 22–24 February 2012; pp. 89–94.
34. Liu, C.; Bouazizi, I.; Gabbouj, M. Rate adaptation for adaptive HTTP streaming. In Proceedings of the 2nd ACM Conference on Multimedia systems, San Jose, CA, USA, 23–25 February 2011; pp. 169–174.
35. Huang, T.Y.; Johari, R.; McKeown, N.; Trunnell, M.; Watson, M. A buffer-based approach to rate adaptation: Evidence from a large video streaming service. In Proceedings of the ACM Conference (SIGCOMM'14), Chicago, IL, USA, 17–22 August 2014; pp. 187–198.
36. Juluri, P.; Tamarapalli, V.; Medhi, D. SARA: Segment aware rate adaptation algorithm for dynamic adaptive streaming over HTTP. In Proceedings of the Workshop on Quality of Experience-based Management for Future Internet Applications and Services, London, UK, 12–15 June 2015; pp. 1765–1770.
37. Timmerer, C.; Maiero, M.; Rainer, B.; Petscharnig, S. Quality of Experience of Adaptive HTTP Streaming in Real-World Environments. *IEEE COMSOC MMTC* **2015**, *15*, 1–4.
38. De Pessemier, T.; De Moor, K.; Joseph, W.; De Marez, L.; Martens, L. Quantifying the influence of rebuffering interruptions on the user's quality of experience during mobile video watching. *IEEE Trans. Broadcast.* **2013**, *59*, 47–61. [CrossRef]
39. IETF RFC-8402. Segment Routing Architecture. Available online: Https://tools.ietf.org/html/rfc8402 (accessed on 10 January 2019).
40. Kakhki, A.M.; Jero, S.; Choffnes, S.; Nita-Rotaru, C.; Mislove, A. Taking a Long Look at QUIC: An Approach for Rigorous Evaluation of Rapidly Evolving Transport Protocols. In Proceedings of the 2017 Internet Measurement Conference, London, UK, 1–3 November 2017; pp. 290–303.

41. Liotou, E.; Samdanis, K.; Pateromichelakis, E.; Passas, N.; Merakos, L. QoE-SDN APP: A Rate-guided QoE-aware SDN-APP for HTTP Adaptive Video Streaming. *IEEE J. Sel. Areas Commun.* **2018**, *36*, 598–615. [CrossRef]
42. De Coninck, Q.; Bonaventure, O. Multipath QUIC: Design and Evaluation. In Proceedings of the International Conference on Emerging Networking EXperiments and Technologies (Conext17), Incheon, Korea, 12–15 December 2017.