On Maximizing QoE in AVC-Based HTTP Adaptive Streaming: An SDN Approach

Alireza Erfanian, Farzad Tashtarian
Department of Computer Engineering
Mashhad Branch, Islamic Azad University, Mashhad, Iran
Email: {a.erfanian},{f.tashtarian}@mshdiau.ac.ir

Mohammad Hossein Yaghmaee
Department of Computer Engineering
Ferdowsi University of Mashhad
Email: hyaghmae@um.ac.ir

***Abstract*—HTTP adaptive streaming (HAS) is quickly becoming the dominant video delivery technique for adaptive streaming over the Internet. Still considered as its primary challenges are determining the optimal rate adaptation and improving both the quality of experience (QoE) and QoE-fairness. Recent studies have shown that techniques providing a comprehensive and central view of the network resources can lead to greater gains in performance. By leveraging software defined networking (SDN), the current study proposes an SDN-based approach to maximize QoE metrics and QoE-fairness in AVC-based HTTP adaptive streaming. The proposed approach determines both the optimal adaptation and data paths for delivering the requested video files from HTTP-media servers to DASH clients. In fact, the proposed approach, which includes a set of application modules, is centrally executed by an SND controller in a time slot fashion. We formulate the problem as a mixed integer linear programming (MILP) optimization model in such a way that it applies defined policies, e.g. setting priorities for clients in obtaining video quality. We conduct experiments by emulating the proposed framework in Mininet using Floodlight as the SDN controller. In terms of improving QoE-fairness and QoE metrics, the effectiveness of the proposed approach is validated by a comparison with different approaches.**

***Index Terms*—Dynamic HTTP Adaptive Streaming (DASH), Software defined networking (SDN), QoE.**

I. Introduction

In recent years, the presence of the Internet and its applications in various aspects of our lives has exponentially increased. As mentioned in [1], video streaming traffic, which has made up the largest portion of Internet traffic, will grow to 75% of the total Internet traffic by 2020. For many years, the UDP protocol has been employed to transfer multimedia traffic in the Internet. In recent years, much effort has been spent to utilize TCP for multimedia transmission over the Internet. With HTTP, the employment of caches and also content delivery networks (CDNs) are possible, thus providing network scalability and traffic reduction. In addition to TCP's reliable transmission and cache-friendliness, streaming data with TCP allows for easy traverse of firewalls and NAT devices. As a result, deploying HTTP for multimedia transmission has significantly risen. For instance, nowadays, more than 98% of video traffic in cellular networks is transmitting via HTTP protocol [2].

978-1-5386-2542-2/18/31.00 © 2018 $IEEE$

The most widely used technique for TCP streaming is HTTP-based adaptive streaming (HAS). In this technique, a video file is divided into short duration segmented files, each of which is encoded at different bit rate levels and resolutions. Many companies have developed modifications of HAS systems, such as Smooth Streaming [3], HDS [4], and HLS [5]. In 2012, HAS was standardized by the motion picture experts group (MPEG) and named dynamic adaptive streaming over HTTP (DASH) [6]. With DASH, a video is divided into short segments encoded at various bit rates. This information is then stored in a media presentation description (MPD) file.

There are several encoding methods for encoding the video such as advanced video coding (AVC) [7]. In AVC, Multiple copies of a video are encoded with different bit rates and stored on the media server. In AVC-DASH, the client downloads the MPD file to obtain information from the server, such as segment details, available bit rates, etc. Then, the appropriate bit rate is selected and HTTP streams these segments to the end-users. In general, maximizing QoE and QoE-fairness can be achieved by employing an optimal adaptation. This can be performed by three approaches: 1) *Purely client-based*, 2) *Client-based assisted by network elements*, and 3) *Network-based*. In purely client-based, according to the local parameters (e.g., available bandwidth and buffer occupancy), the client adapts the video quality level. However, since clients are unaware of the whole network topology and its current state, this adaptation technique is sub-optimal and so can be disadvantageous in shared environments. In the case of client-based assisted by network elements, client adaptation can be assisted by an element in the network, such as proxy servers. In Network-based, by utilizing the complete view of the current network state, rate adaptation can be performed by a centralized controller [8]. A comprehensive investigation on these methods is presented in the next part. Software defined networking (SDN) has recently emerged as a networking paradigm. In this architecture, the data and control plane are decoupled, which sheds new light on networking technology [9], [10]. The data plane consists of hardware or software elements dedicated to forwarding flows and packets. In contrast, with an SDN controller, the control plane manages the data plane elements. By applying the current study's innovations to the network, we can develop network applications in the control plane that are able to communicate with the SDN

controller through application programming interfaces (APIs), e.g, RESTful API. In fact, SDN provides a flexible platform which can perform adaptive routing algorithms for different network applications. In addition, SDN can modify traffic to increase QoE for certain traffic flows (e.g., multimedia streaming traffic).

In the present work, we propose an SDN-based approach to maximize the QoE and QoE-fairness of HTTP adaptive streaming. In this approach, by introducing a set of connected application modules, the SDN controller centrally adjust the DASH clients' adaptation rate. In fact, by employing a holistic network view provided by the SDN controller and collecting some critical information from HTTP-media servers and DASH clients, the SDN controller jointly determines the appropriate quality adaptation and flow paths for the client requests. We firstly define substantial QoE metrics and QoE-fairness in the process of designing and elaborating the proposed architecture which consists of a set of application modules. Then the addressed problem in form of a mixed integer linear programming (MILP) model, is illustrated which is NP-complete. Finally, for comprehensive evaluation, we implement the proposed approach and compare with some similar studies.

The remainder of this paper is organized as follows. In Part II, related work is presented. Part III introduces and elaborates on the proposed approach and its details. The performance evaluation is presented in Part IV. Finally, Part V concludes the paper.

II. Related Work

In this section, we discuss the three main solutions that exist to determine the optimal adaptation which are *purely client-based*, *client-based assisted by network elements* and *network-based* adaptation techniques. In the *purely client-based* technique, a client performs quality adaptation based on its local parameters, such as network throughput, occupied buffer size, etc. In the second approach, these clients can be assisted by the information provided by network elements, such as proxy servers. In the third methods, we use a central network element to perform rate adaptation on behalf of the clients.

A. Purely Client-Based Adaptation

As shown in [11], [12] DASH video players in clients are responsible for quality adaptation in order to optimize the QoE objectives, e.g., initial buffering time minimization, stalling minimization, and quality maximization. In [13], [14], the authors propose a general bit rate adaptation framework that consists of a set of methods striving to achieve a trade-off between video stability, fairness, and efficiency. A stateful bit rate selection heuristic algorithm is used to achieve a biased interaction between the bit rate and estimated bandwidth. However, this client-based quality adaptation leads to uneven bandwidth competition, which intensifies when a large number of clients use the shared network resources [15]. Moreover, resource fairness does not produce QoE-fairness, specially in heterogeneous environment [16]. This unfair resource usage, coupled with unpredicted network traffic bursts, can cause frequent changes in the perceived video quality in clients (client quality oscillations), which significantly reduces QoE. Therefore, an optimization model should be developed to trade-off between different QoE parameters, such as the video quality maximization and minimization of bit rate changes, while also meeting network constraints.

In order to achieve this goal, various algorithms have been developed through the consideration of different parameters, such as estimated network bandwidth, network throughput, and received network feedback signals (e.g., congestion occurrence) [17], [18]. However, it is clear that, by using purely client-side sub-optimal adaptation decisions and with the absence of a central controller for clients, the video quality changes frequently and QoE reduction ultimately occurs [11], [19]. Furthermore, with a central controller, it is possible to enforce different management policies, such as various subscription policies (e.g., Gold, Silver, Bronze) [20].

B. Client-Based Adaptation Assisted by Network Elements

As we mentioned before, specific network elements can assist clients for client-based rate adaptation. Using the information provided by network elements, a client can enhance its quality adaptation procedure [8]. Utilization an SDN controller with a holistic network view, assist client-side quality adaptation in an AVC-DASH streaming method. Additionally, in [21], the authors propose an SDN-based video streaming approach to fairly maximize the QoE of multiple competing clients in a shared network environment. Likewise, the proposed strategy in [22] is based on SDN. In fact, the authors of [22], employ two main approaches to optimize QoE, in terms of the number of quality changes and fairness. In the first approach, the SDN controller determines the video quality of the clients. In the second one, a queue for each client is developed to provide dynamic rate adaptations. Although these studies utilize some network elements to improve QoE, the determination of optimal adaptations is performed separately by clients. Thus, these strategies do not lead to efficient share usage of network resources. Bentaleb *et. al.* in [15] address HAS scalability issues, including video instability, QoE-unfairness, and network resource under-utilization. To cope with these issues, they maximize the QoE per client. Moreover, [16] mitigates the main drawbacks of SDNDASH[15]: scalability , communication overhead, and the support of client heterogeneity.

There are some works, however, that have mainly focused on QoS-aware video traffic routing [23], [24], [25]. In fact, these studies have investigated the QoS-aware video flow routing in OpenFlow/SDN-enabled networks. Thereafter, by considering the QoS priorities; the routing algorithm finds the shortest path between the media server and client. In addition, Egilmez *et. al.* [24] design a QoS-aware controller to deliver multimedia flows in OpenFlow-enabled networks. In this approach, they classify input traffic based on data flow types. Then, multimedia flows are routed through QoS-guaranteed

paths and other flows are delivered to the destination via the shortest paths. These studies do not discuss quality adaptation techniques. Although these techniques can improve client-side decisions, these decisions are still sub-optimal as there is no independent central element for decision making. Furthermore, clients might not be able to perform the decisions made by the controller [22]. Therefore, central-based quality adaptation can be suggested.

C. Network-Based Adaptation

Several works use a proxy/controller as a central rate adaptation decision-making authority [26], [27], [28]. In [26], the authors propose an approach that actually employs a proxy server to monitor video quality requests. In this approach, a proxy server periodically solves an optimization problem in order to determine the maximum segment quality levels which the clients can download. This is achieved based on the current network status and a specific objective function. In fact, whenever a client-based adaptation decreases overall fairness, the proxy server is able to replace its video quality decision with that of the client. Therefore, a certain level of QoE can be guaranteed among some or all clients. Moreover, Mok *et. al.* [27] design a proxy architecture to prevent frequent and drastic oscillations and to allow a specific amount of change in video quality in each step. Unlike ours, this approach is not scalable, since all the network traffic to the video server must pass through a proxy server. Furthermore, [28] presents reactive and proactive QoE optimization approaches. In the reactive method, client-based quality adaptation is achieved using network information. By considering client buffer utilization and quality adaptation in the controller, the second method provides a higher and fairer video quality to clients.

III. The Proposed Approach

In this section, by leveraging the SDN conceptual model, the current study addresses the issues of client quality adaptation, quality of experience(QoE) and QoE-fairness. Before introducing the details of the proposed approach, we shall describe the considered QoE and the QoE-fairness metrics as follows:

- *Startup delay*: The time period between sending a video request and starting to render the first frame.
- *Number of Stalls*: The stall phenomenon occurs when the buffer space of a client reaches to its minimum threshold value (may be zero) during playback of the requested video. At this time, a DASH player stops rendering and waits to download the current segment.
- *Average video quality*: The average of the video quality of all received segments.
- *Average number of video quality switches*: This metric is expressed as the average numbers of video quality switches between any successive received segments.
- *QoE-fairness*: The variance of video quality received by clients at the same time.

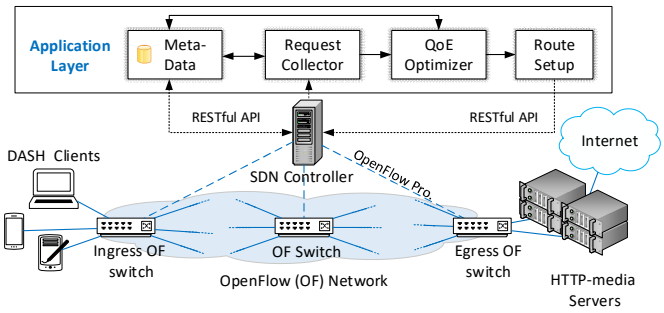

Figure 1. **The communication topology of proposed approach**

A. The details of proposed approach

As mentioned earlier, a DASH player/client first downloads the media presentation description (MPD) file that provides the segment information and then requests each segment individually. However, we assume that the DASH clients can determine and send the maximum supported video quality level, according to their resource capacity. In this study, the time for buffering a segment is considered fixed, even though it is possible that a DASH player suggests a deadline in which the requested segment must be downloaded.

The schematic communication topology of the proposed approach is illustrated in Fig. 1. As depicted, the HTTP-media servers can serve DASH clients through the underlying OpenFlow (OF) network that is monitored and managed by the SDN controller. In fact, the proposed approach that consists of four application modules (request collector, QoE optimizer, route setup, and meta-data), operates in a time slotted manner, in which, in each time slot τ, after gathering the requests of DASH clients an optimal solution is determined and the OF switches are configured accordingly (See Fig. 2).

The ingress OF switch (the first hop OF switch of the DASH client) needs to distinguish requested media packets, in order to forward them to the SDN controller. Although deep packet inspection(DPI) is the common approach to examine IP packets at the Application layer of the OSI model [29], to evade DPI overhead, we suppose that DASH clients are configured to connect a specific IP and port address as a *default HTTP-media server*. Therefore, an ingress OF switch receiving a packet with destination IP and the port address of the default HTTP-media server, is configured to be forwarded it to the SDN controller as a Packet-In. The request collector module is responsible to gather all requests in each time slot, and triggers QoE optimizer module to determine an optimal solution. Thereafter, according to the output of route setup module, the SDN controller configures OF switches.

This communication model can be utilized in the network edge, and the requests of DASH clients will be served by HTTP-media cache servers if the requests hit, else they must be forwarded to the origin server located on the Internet. However, in this study, we assume that all DASH clients' requests can be served by the local HTTP-media servers. In following, we describe the functionality of each application

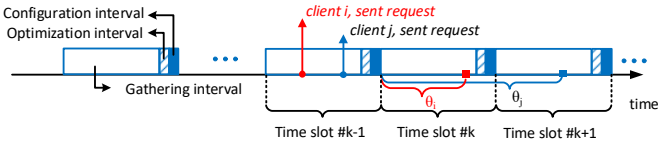

Figure 2. **The time slot in the proposed approach**

module:

Request Collector Module (RCM): After receiving the Packet-In, the SDN controller forwards it to the RCM. RCM is responsible for receiving the DASH client requests at each time slot τ, at which point it launches the QoE optimizer module. In fact, each time slot τ is divided into three unequal intervals: the gathering, optimization, and configuration interval (see Fig. 2). The two short optimization and configuration intervals are allocated for running the QoE optimizer module and configuring OF switches to start data transmission, respectively. According to the output of the QoE optimizer module, it is possible that data transmission takes longer than the time slot. In the gathering interval, RCM buffers the received requests from the DASH clients and performs some pre-processing on the collected requests so as to prepare them as an input parameter for the QoE optimizer module in the next time slot.

In the pre-processing step, RCM must extract some key values from the client requests. According to the received requests in each gathering interval, RCM creates two sets, namely $\mathbb{C}$ and $\mathbb{D}$. The first set, $\mathbb{C}$, includes the ingress OF switches with their connected DASH clients, where $\mathcal{N}_i$ indicates the DASH clients connected to the ingress OF switche i. $\mathbb{D}$ is the set of DASH clients requesting to download desired quality of demanded segments. Since the QoE optimizer module needs to take all gathered requests into account, RCM assigns a unique identification (ID) for each request by parsing the headers of transport and the application layers of the incoming packets, e.g. a combination of incoming sockets and the file name of the requested segment. Furthermore, RCM suggests θ_c as the deadline for buffering the requested segment (the waiting time of DASH client $c \in \mathbb{D}$ in the gathering interval can be taken into account) and m_c as the maximum video quality level of the segment determined by client $c \in \mathbb{D}$. We note here that the values of θ_c can be assumed as fixed, However, to achieve a more flexible model, it is possible these be determined by DASH players or RCM as appropriate. As illustrated in Fig. 2, the two clients, i and j, send their requests in time slot $k-1$. Both clients must wait for the optimization interval of the next time slot (time slot k) to obtain an optimal solution. As shown, the deadlines for the requested video files in time slot $k-1$ are considered as being from the beginning of the next time slot or time slot k.

Meta-data Module (MM): Before describing MM, let us define graph $G = \{V, E\}$, where V is the set of OF switches, DASH clients, and HTTP-media servers; and set E represents the edges of G, where $e_{ij} = 1$ if a direct communication link exists between the two entities of i and $j \in V$. At the beginning of each time slot, MM uses RESTful APIs to obtain the available links' bandwidth from the SDN controller. The measured values are stored in a two-dimensional array B, in which b_{ij} indicates the available bandwidth between i and $j \in V$. The connectivity among OF switches, DASH clients, and HTTP-media servers can easily be inferred from B. In fact, this module constructs graph $G = \{V, E\}$ and the available bandwidth among the nodes in V.

Moreover, this module provides critical meta-data about the available video files in HTTP-media servers that includes the list of servers, stored file segments, qualities, and their properties, such as the size and bit-rate of each quality. We define $\mathbb{S}$ as the set of HTTP-media servers, in which at least one quality level q requested by DASH client $c \in \mathbb{D}$ is stored, where $q \leq m_c$. In fact, the data catalog module sets $a^s_{cq} = 1$ if quality q requested by c is available in server s, otherwise $a^s_{cq} = 0$. The data catalog module also uses δ_{cq} to show the size of quality level q requested by DASH client $c \in \mathbb{D}$.

QoE Optimizer Module (QOM): In this module, we introduce a mixed integer linear programming model (MILP) which, by maximizing the QoE-fairness and QoE jointly determines the optimal quality adaptation and data paths for delivering the requested data from HTTP-media servers to DASH clients. To find an optimal solution, a number of constraints must be satisfied. Before describing these, we shall define the input parameters and variables as follows:

Input parameters:

$-G(V, E)$: represents the network topology.

$-\mathbb{D}, \mathbb{C}, \mathbb{S}$: are sets of DASH clients, ingress OF switches and HTTP-media servers, respectively.

$-B(b_{ij})$: is a two-dimensional array where b_{ij} shows the available bandwidth between i and $j \in V$.

$-A(a^s_{cq})$: is a three-dimensional binary array where $a^s_{cq} = 1$ represents that quality level q requested by $c \in \mathbb{D}$ is reachable through $s \in \mathbb{S}$.

$-\theta_c$: is an offered deadline for delivering the requested segment from HTTP-media server to $c \in \mathbb{D}$.

$-m_c$: is the maximum supported quality level determined by $c \in \mathbb{D}$.

$-\delta_{cq}$: is the size of quality level q requested by DASH client c.

Variables:

$-t^{cq}_{ij}$: is an optimal data rate for transmitting the requested quality level q by client c that must be delivered from i to $j \in V$

$-\omega^s_{cq}$: is a binary variable determines that whether the requested quality level q by client c must be served by server s ($\omega^s_{cq} = 1$) or not.

$-T_c$: is an average video quality level of the client c from the beginning of its work until now.

$-N_c$: is an average number of quality level switches for $c \in \mathbb{D}$.

$-\nu_c$: is a binary variable that determines whether the quality oscillation of received segment happens ($\nu_c = 1$) or not.

$-Q$: determines the biggest gap between supported maximum quality m_c and the maximum of served quality by the network for each client c.

The MILP used in QOM can be represented as follows:

minimize

$$\alpha Q + \frac{1}{len(\mathbb{D})}\sum_{c\in\mathbb{D}}(\beta_{1c}N_c - \beta_{2c}T_c) + \epsilon\sum_{c\in\mathbb{D}}\sum_{q=1:m_c}\sum_{i,j\in V} t_{ij}^{cq} \tag{1}$$

s.t.

$$\sum_{s\in\mathbb{S}}\sum_{q=1:m_c}\omega_{cq}^{s}\times a_{cq}^{s} = 1 \qquad \forall\ c\in\mathbb{D}\ (I)$$

$$\theta_c(\sum_{j\in V} e_{ij}\times t_{ij}^{cq} - \sum_{j\in V} e_{ji}\times t_{ji}^{cq}) = \qquad (II)$$

$$\begin{cases} \delta_{cq}\times\omega_{cq}^{i} & \forall\ i\in\mathbb{S},\ c\in\mathbb{D},\ q=1:m_c \\ 0 & \forall\ i\in V-\{\mathbb{S},\mathbb{C}\},\ c\in\mathbb{D},\ q=1:m_c \\ -\delta_{cq}\sum_{s\in\mathbb{S}}\omega_{cq}^{s} & \forall\ i\in\mathbb{C},\ c\in\mathcal{N}_i,\ q=1:m_c \end{cases}$$

$$\sum_{c\in\mathbb{D}}\sum_{q=1:m_c} e_{ij}\times t_{ij}^{cq} \le b_{ij}, \qquad \forall\ i,j\in V\ (III)$$

$$m_c - \sum_{s\in\mathbb{S}}\sum_{q=1:m_c}\omega_{cq}^{s}\times q \le Q\times\ m_c, \qquad \forall\ c\in\mathbb{D}(IV)$$

$$\frac{1}{\varphi_c}(\bar{\lambda}_c + \sum_{s\in\mathbb{S}}\sum_{q=1:m_c}\omega_{cq}^{s}\times q) = T_c\times T_{max},\ \forall\ c\in\mathbb{D}\ (V)$$

$$|(\sum_{s\in\mathbb{S}}\sum_{q=1:m_c}\omega_{cq}^{s}\times q) - \bar{l}_c| \le \nu_c m_c, \qquad \forall\ c\in\mathbb{D}\ (VI)$$

$$\frac{1}{\varphi_c}(\bar{\nu}_c + \nu_c) \le N_c N_{max}, \qquad \forall\ c\in\mathbb{D}\ (VII)$$

vars. $Q, T_c, N_c \in [0,1], t_{ij}^{cq}\ge 0,$ and $\omega_{cq}^{s}, \nu_c \in \{0,1\}$

The first constraint states that one server can serve each requested quality level by client c. Constraint (II) determines data rates for transmitting requested quality level from optimal selected HTTP-media server in $\mathbb{S}$ to the DASH clients in $\mathbb{D}$. This constraint covers three different cases: if i is an HTTP-media server ($i\in\mathbb{S}$), it forces server i to generate $\frac{1}{\theta_c}\delta_{cq}$ amount of the traffic. As the second case, if i is an OF switch, then $\forall c\in\mathbb{D}$ and $q=1:m_c$, the total incoming traffic to it must be equal to the total outgoing traffic from it. Finally, the last case states that the all data of quality level q that are transmitting from server s to the client c must be received by the ingress OF switch i (if c connects to it). The next constraint specifies an upper bound for the generated traffic on each link.

To consider the QoE-fairness and QoE metrics in our proposed MILP model, a number of constraints should be met. First, we shall focus on the QoE-fairness in serving DASH clients in each time slot. To have a fair video quality level regarding the different maximum supported video quality level by clients, we represent constraint (IV) to determines the biggest gap, denoted by $0\le Q\le 1$, between the supported maximum quality level m_c and the maximum quality level served by the network for each client c. It is obvious that, by decreasing Q in the objective function, the variance of the video quality level received by clients will reduce and consequently the QoE-fairness will increase.

We note that the first two QoE metrics, *start up delay* and *number of stalls*, are taken into account by selecting the appropriate value for θ_c. In fact, θ_c forces the MILP model to forward the requested quality level through determining the optimal data rates (See constraint (II)). However, for the *number of video quality switches*, we need to keep track of their treatments in prior time slots. Let $\bar{\lambda}_c$ be the total video quality level of the segments downloaded by client c from the time its first request is sent until the current time slot. Thus, T_c is obtained through constraint (V), where $T_{max} = max\{\bar{\lambda}_i + m_i | \forall i\in\mathbb{D}\}$ and φ_c and $\sum_{s\in\mathbb{S}}\sum_{q=1:m_c}\omega_{cq}^{s}\times q)$ are the total number of requested segments and the achieved quality level of the video requested in the current time slot by client c, respectively. Note that time slot oscillation can be determined with respect to the perceived quality in the previous time slot. This statement is represented by constraint (VI). Therefore, if $\nu_c = 1$, the quality level of the video oscillates else it remains unchanged. In constraint (VI), $|\sum_{s\in\mathbb{S}}\sum_{q=1:m_c}\omega_{cq}^{s} - \bar{l}_c|$ shows the difference in video quality level among the quality level obtained in the previous time slot denoted by $\bar{l}_c$, and the achieved quality level in the current time slot, as well as the notation $|.|$ indicates the absolute operation. Let $\bar{\nu}_c$ be the number of video quality level switches by DASH client c while obtaining segments from the beginning of its operation until the current time slot. Thus, the normalized average number of video quality switches, N_c, for client c is obtained through the last constraint where $N_{max} = max\{\bar{\nu}_i + 1 | \forall i\in\mathbb{D}\}$.

In each time slot, QOM runs the above MILP model to maximize both QoE metrics and QoE-fairness by determining the optimal quality adaptation and data rates to transmit the requested quality level. In addition to maximizing the defined QoE metrics and QoE-fairness, the objective function Eq. (1) implicitly preventing the waste of network resources by decreasing the total generated traffic (the last term in the objective function). Moreover, we can also set desirable priorities for these metrics using constant weights $\alpha, \beta_{1c}, \beta_{2c}$, and ϵ. For instance, β_{1c} can assist us to determine how important client c is to receive segments without any quality oscillation. The weights can be adjusted by the application policies in each time slot.

Theorem 1: The proposed MILP formulation (1) is an NP-complete problem.

Proof. Let us consider the most basic form of the problem as follows: to maximize the sum of video qualities, it is assumed that all HTTP-media servers are connected to the ingress OF switches through a link with limited bandwidth and that each HTTP-media server can serve requested level as well. Now, by defining the weight and value for each video level stored in the HTTP-media servers as the required bandwidth and the perceived video quality level, then the problem can be reduced in a polynomial time to the classic knapsack problem, in which the maximum knapsack capacity equals the available bandwidth. Moreover, by considering the fairness, the problem can be reduced to the quadratic knapsack problem, in which opting any two equal levels, in terms of video quality, results in more benefits. □

Route Setup Module (RSM): After determining the sub-optimal solution by QOM, we need to utilize a method to support a variety of data transmission rates in the network

through accurate configurations of OF switches and HTTP-media servers. In fact, by sending RESTful API commands to the SDN controller, RSM performs the final step in launching data flows between HTTP-media servers and DASH clients. Since OF switches support queues so as to provide desirable data rate calibration on each flow [30], configuring different queues on all OF switches may be a simple yet naive idea, as there are numerous video files with different data rates that must be delivered in each time slot. In addition, the probability of misconfiguration grows as the number of switches increases. Hence, we propose an agile and reliable method based on tagging the *type of service* (TOS) field of the data packets originated by the HTTP-media servers to the destination DASH clients. Generally, the tagging process can be performed by HTTP-media servers or egress OF switches. However, to achieve a minimum overhead in OF switches, we delegate the tagging process to the HTTP-media servers.

The RSM operation for each requested segment can be divided into two main sub-steps: (1) configuring the HTTP-media server to tag the outgoing packets and (2) configuring OF switches to support the determined data paths and rates.

Sub-step 1: In this sub-step, RSM should determine the minimum required tags for generated data packets and then configure the selected HTTP-media server to apply the tags. For ease of explanation, let us present an example illustrated in Fig.3. Suppose all DASH clients are configured to send their request to a default HTTP-media server with IP='X' and port='Y'. Thus, as illustrated in Fig. 3 (a), ingress OF switch '$v2$' receives a request from DASH client 'B' with destination IP 'X' and port 'Y' and sends this to the SDN controller in the form of a PACKET-IN message. Upon receiving the PACKET-IN message, the SDN controller delivers it to RCM. After processing the request received by RCM, QOM selects the HTTP-media server 'A' to serve DASH client 'B' for quality level 'q' (size 100kB) according to the output of QOM (see Fig.3 (b) for the given deadline $\theta_B = 1$ second). Now, with having the optimal data path and rate t_{ij}^{Bq} , the minimum number of tags must be calculated by RSM in this sub-step. In fact, a list of *base* data rates $\mathbb{B}_{Bq} = \{b_1, b_2, ..., b_n\}$ must be produced, where $\forall i, j \in V$, t_{ij}^{Bq} is decomposing into $\mathbb{B}_{Bq}$. To specify $\mathbb{B}_{Bq}$, we propose a simple recursive function named *base_data_rates* (see Algorithm 2). The algorithm traverses the data path from the ingress OF switch '$v2$' to the HTTP-media server 'A' by calling function $base\_data\_rates$ ('$v2$' , 'A'). This function stops calling itself when it meets the data path root (HTTP-media server 'A'). In this example, the obtained base data rates is $\{100kbps, 300kbps, 400kbps\}$ (See HTTP-media server 'A' in Fig.3 (c)). After the base data rates are specified by proposed Algorithm 1, RSM should accordingly configure the HTTP-media server to tag the outgoing packets. To do this, RSM concatenates the following parameters to the URL address in the request packet and then orders the SDN controller to send them back to the ingress OF switch in the form of a PACKET-OUT message: (1) base data rates, (2) deadline value θ_B, (3) list of required tags, and (4) list of egress OF switches connected to the selected HTTP-media server. Note that the destination address of the PACKET-OUT message is set to the selected HTTP-media server. Note that the ingress OF switch forwards the packet to the specified HTTP-media server. As depicted in Fig.3, for the obtained base data rates= $\{100kbps, 300kbps, 400kbps\}$, the RSM sets $tag = \{1, 2, 3\}$ and $egress\ OF\ switches = \{v1, v1, v1\}$. This configuration signifies that the HTTP-media server 'A' must send the amount of $\theta_B \times 100kbps$ data to egress OF switch $v1$ with tag 1, the amount of $\theta_B \times 300kbps$ data to $v1$ with tag 2, and the amount of $\theta_B \times 400kbps$ data to $v1$ with tag 3.

Sub-step 2: In the second sub-step, RSM installs required rules in OF switches to forward packets according to the determined data path. Each rule filters incoming packets according to ToS tag, source and destination IP and port address and then specifies the output port. For more transparency, the ingress OF switch replaces the source IP and port address of the incoming data packet from the HTTP-media server with the default IP and port address.

Algorithm 1: update_data_rate_I(l, $\bar{L}$)

```
l2 = find_link(min_rate(L̄));
set l2 as visited;
r=data_rate(l2);
while r < data_rate(l) do
    delete_link(L̄,l2);
    l2=find_link(min_rate(L̄));
    set l2 as visited;
    r = r+ data_rate(l2);
end
[p,q]=peers(l2);
vl= add_virtual_link(p,q);
set_data_rate(vl, r−data_rate(l));
update_data_rate(l2, data_rate(l2) − (r− data_rate(l)));
```

Algorithm 2: base_data_rates(n, m)

```
//m=server 'A' and in the first call n= ingress switch;
if n == m then
    return;
end
L=all_incoming_link(n);
foreach l in L do
    [n,p] = peers(l);
    // where n and p are two nodes connected by link l;
    L̄ = unvisited_incoming_link(p);
    if data_rate(l) = any_combination(data_rate(L̄)) then
        set the true combination of L̄ as visited;
        continue;
    end
    if data_rate(l) > max_rate(L̄) then
        Call update_data_rate_I(l, L̄);
    else
        Call update_data_rate_II(l, L̄);
    end
    U = upstream_OFswitch(n);
    foreach u in U do
        Call base_data_rates(u, m);
    end
end
```

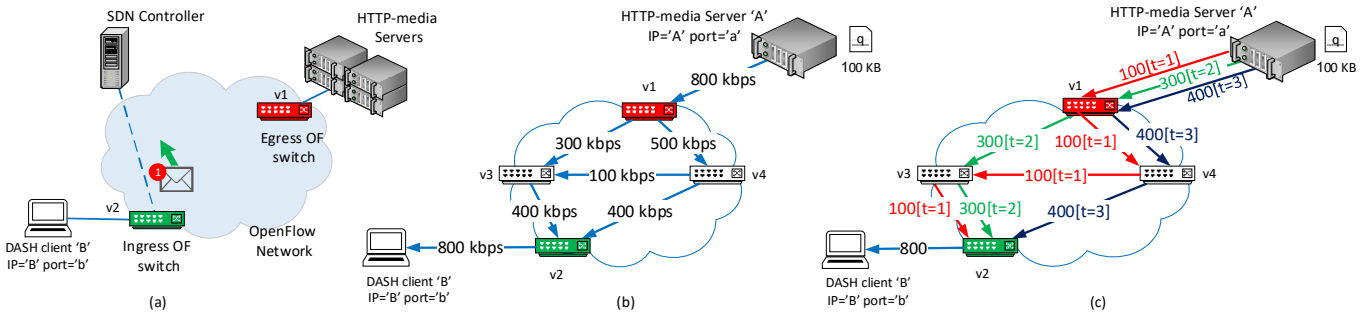

Figure 3. **An example of data transmission between a DASH client and HTTP-media servers: (a) the ingress OF switch sends the request of DASH client to the SDN controller, (b) the determined data paths and rates by QOM , (c) obtaining the desired tags and configuring OF switches.**

IV. Performance Evaluation

In this part, we empirically evaluate and present the performance results of the proposed model. In the following, we first investigate the behavior of the proposed MILP model and various QoE parameters. Then, we continue our investigation by comparing the performance of proposed QoE-fairness method with other algorithms.

For evaluation purposes, we use the *Big Buck Bunny* video provided by [31]. We use the 4 seconds encoded video in three resolutions (1080p, 720p, 360p) and several bitrates as shown in table I. All bitrates used for video encoding and the resulting video qualities measured with SSIM [32] are shown in table I. The structural similarity (SSIM) index is a method for predicting the perceived quality of requested video. The created files are uploaded into simulated web servers in Mininet [33]. The base topology used in the implementation is illustrated in Fig.4. There are seven OF switches and six clients with different resolution connected to the system via two ingress OF switches. As seen in Fig.4, the bandwidth between the OF switches is set at 8 Mbps. In fact, to simulate a bandwidth-limited network and to investigate the treatment of the framework, we limit the core network bandwidth capacity (i.e. switch to switch). In other hand in our setup, clients join the network in predefined random time-slots within the first 15s. The other initial parameter values are set as follows: $\theta_c = 1.5s, \tau = 1s, \alpha = 0.4, \beta_{1c} = 0.2, \beta_{2c} = 0.4$ and $\epsilon = 0.00001, (for\ \text{c=1} : 6)$. We use Mininet to generate the network topology and perform the evaluations. We also employ Floodlight [34] as the SDN controller in our experiments. To emulate the clients' video player, we use the *Scootplayer* [35] that is an experimental MPEG-DASH request engine which also provides accurate logs. In next, we shall evaluate the behavior of the system by changing the values of different parameters.

Algorithm 3: update_data_rate_II(l, $\bar{L}$)

```
l2 = find_link(max_rate(L̄));
[p,q] = peers (l2);
vl = add_virtual_link(p,q);
set vl as visited;
set_data_rate(vl,data_rate(l));
update_data_rate(l2, data_rate(l2)−data_rate(l));
```

A. QoE Parameters

According to the buffering deadline 'θ_c' and the requested segment duration, we can eliminate the stalling phenomenon in clients by setting $\theta_c \leq (segment\_duration) - (time\_slot\_duration\ \tau)$. The transmission delay can also be considered when determining the θ_c value. Thus, by choosing $\theta_c \leq 3s$, model can guarantee that stalling phenomenon never happened. The next QoE parameter, startup delay, depends on τ and θ_c too. In fact, in the worst case, a client may join the network exactly at the beginning of when the optimization algorithm is run. In this case, the client has to wait for $\tau + \theta_c$ seconds to receive the video. On the other hand, in the best case, the client's request is submitted exactly before the optimization algorithm is run and the video is delivered after θ_c seconds. Notably, the average waiting time for clients may be expressed as $\theta_c + \frac{\tau}{2}$. The empirical results proved these statements.

In the first experiment, we focus on the video quality level received by clients. To measure this parameter, we use the structural similarity metric (SSIM). In fact, SSIM can be used to assess the mean delivered video quality to clients in each time slot (Fig. 5(b)).

In the next experiment, we investigate the impact of changing θ_c in the video delivered quality level. As a reminder, θ_c is defined as the video delivery deadline for client c. θ_c is assumed to be identical for all clients. Obviously, by increasing the value of θ_c, resource utilization can decrease and a higher video quality may be delivered to clients. This is because the proposed model has more time to deliver the requested video

Table I
The bit rate of video quality levels

Resol.	Video quality levels(kbps)				SSIM index			
360p	180	220	325	380	.95	.957	.964	.968
720p	782	1010	1210	1475	.949	.958	.963	.968
1080p	2088	2410	2950	3340	.95	.957	.963	.969

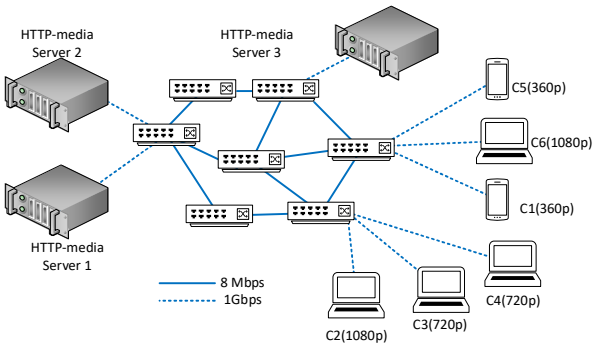

Figure 4. **The considered network topology**

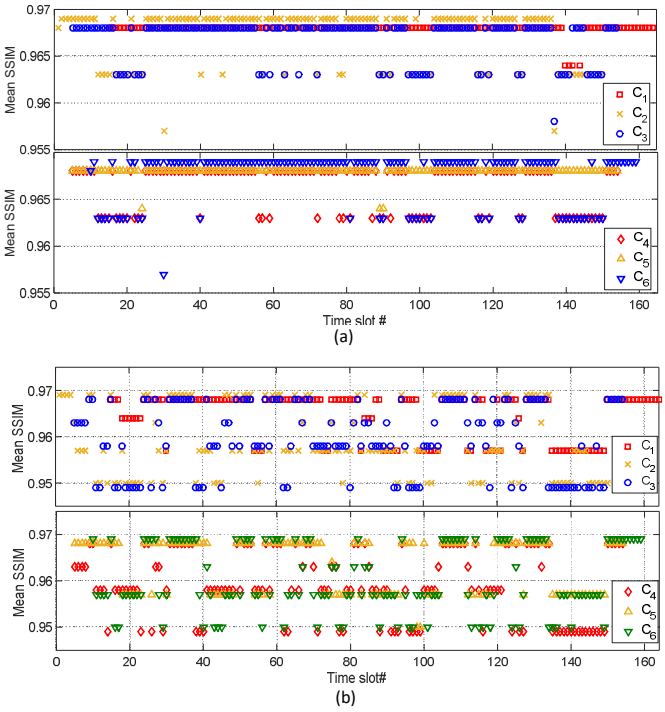

Figure 5. **The impact of changing θ_c value: (a)$\theta_c = 1.5s$,(b)$\theta_c = 1s$**

and consequently it is able to send higher video quality level at a lower transmission data rate. The treatments of the model for different values of θ_c are illustrated in Fig.5. The normal status of the network ($\theta_c = 1.5$) is shown in Fig.5 (a). By increasing the θ_c value from 1.5s to 2s, the clients can get the maximum video quality level (i.e. m_c) (due to lack of space, we ignored to illustrate it). Although raising θ_c promotes the overall quality of delivered video, the possibility of stalling occurring for clients grows. In fact, the stalling phenomenon specifies an upper bound for θ_c. Basically, by considering the buffer level in client c, the proposed framework can centrally offer an appropriate θ_c. We will elaborate on this in a future study. Notably, by choosing a smaller value for θ_c (Fig. 5 (b)), we can force the model to immediately fill the buffer of the clients; in this case, the model has to send the highest possible quality to the clients with high data rate. Therefore, as for limited bandwidth, it is obvious that some clients will be deprived of obtaining more quality (see Fig.5 (b) for $\theta_c = 1$). Moreover, as illustrated in Fig.5 (b), the switching of video quality is increased.

B. QoE-Fairness

To investigate video quality fairness parameter, we use QoE-fainess index introduced by Hossfeld et. al. [36] as follows:

$$F_{SSIM} = 1 - \frac{2 \times \sigma_{SSIM}}{\sigma_{Max} - \sigma_{Min}} \tag{2}$$

where σ_{SSIM} denotes standard deviation of SSIM values observed by clients in each time slot. σ_{Max} and σ_{Min} represent the maximum and minimum SSIM values for available videos in HTTP media-servers, respectively. In our model, parameter α is responsible for providing QoE-fairness in the network (see the proposed model (1)). Fig.6 presents the status of the system in default mode ($\alpha = 0.4$). As depicted in Fig.6, the Q value in each time slot, the biggest gap between delivered quality levels, and the measured QoE-fairness share the same trend. We extended our experiments by giving the priority to the QoE-fairness and the delivered video quality level, through setting parameters as shown in table II. Although in the first experiment($\alpha = 0.9$) the average QoE-fairness is less than others, the model performs better in the minimum QoE-fairness, Q average and the variance of Q. This means model delivered fairer quality by prioritizing QoE-fairness parameter. In the other hand, giving priority to the video quality force the model to deliver higher video quality level to clients. As mentioned in table II, in this scenario average QoE-fairness is higher, however, the minimum QoE-fairness is the least. So, by increasing of α the difference between the maximum and minimum QoE-fairness can be decreased and better QoE-fairness is achievable.

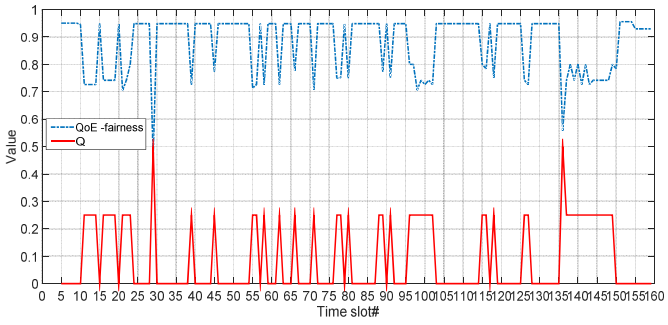

Figure 6. **The QoE-fairness measured based on[36] and the value of Q in the model**

By employing the proposed framework in [37], we compare our proposed MILP model with the QoE-FF framework [21], SPM [38], and NADE [39]. As illustrated in Fig. 7, the proposed model outperforms other methods in terms of start-up delay, while all the studied frameworks report almost all the same results for the min SSIM, mean SSIM, and stalls duration parameters. While the fairness parameter (SSIM) in our proposed model is less than that of NADE, it outperforms

Table II
THE IMPACT OF CHANGING $\alpha, \beta_1, and \beta_2$
IN VARIOUS SCENARIOS

	$\alpha = 0.9$ $\beta_1 = 0.05$ $\beta_2 = 0.05$	$\alpha = 0.4$ $\beta_1 = 0.2$ $\beta_2 = 0.4$	$\alpha = 0.05$ $\beta_1 = 0.05$ $\beta_2 = 0.9$
Average QoE-fairness	0.87545	0.8779	0.8782
Max QoE-fairness	0.9552	0.9552	0.9552
Min QoE-fairness	0.532	0.4655	0.3975
Q average	0.0869	0.0823	0.0843
Q variance	0.0922	0.0897	0.092

the SPM approach. In the quality switch metric, QoE-FF and our proposed model perform much better than do other frameworks.

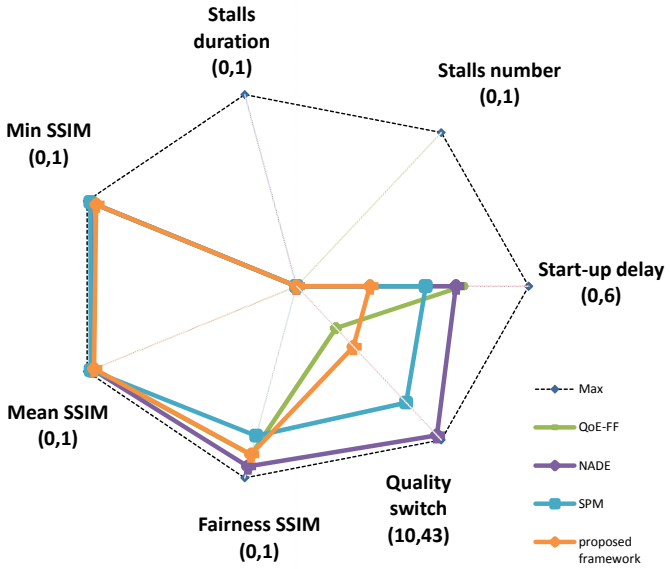

Figure 7. **Compare proposed framework with other frameworks**

V. CONCLUSION

Today, dynamic adaptive streaming over HTTP (DASH) is emerging as the distinguished technology for delivering video over the Internet; it is indebted to the simplicity and the efficacy of HTTP protocol. DASH enables clients to adjust their adaptation regarding the observed feedback from the network. Moreover, DASH provides an opportunity to serve clients by media cache servers at the edge of the network. Delegating an optimal adaptation process to clients has surfaced as an issue where adaptation is performed based on local information on the client. Furthermore, achieving maximum QoE metrics and QoE-fairness cannot be guaranteed by employing local parameters on the clients. By leveraging the SDN paradigm, the present paper proposed a new SDN-based framework to address the problem of rate adaptation by focusing on maximizing QoE and QoE-fairness. We elaborated on the architecture of the proposed model by designing a set of interconnected modules. After collecting critical data from the network, such as client requests and network resource, proposed model determines an optimal data path and rates in a time-slot based manner. In fact, upon receipt of a request packet from the DASH client, it is sent to the SDN controller as a Packet-In. After processing the Packet-In by proposed application modules, an appropriate solution is achieved. Then, the SDN controller configure the OpenFlow switches and HTTP-media servers to start packet transmission. We formulated the problem as a MILP optimization model and showed that it is an NP-complete problem. Regarding the performance evaluation, we implemented the proposed framework and its modules in Python. Then, by employing Mininet and Floodlight, we conducted experiments in different scenarios, evaluated the QoE-fairness and QoE metrics and made comparisons with different state-of-the-art approaches. The results validated the performance of the proposed framework. Extending our proposed framework for live multicast streaming in the network edge and redesign our proposed mathematical model to support HTTP 2.0 can be suggested as two interesting future work directions.

REFERENCES

[1] C. V. N. Index, "Cisco visual networking index: global mobile data traffic forecast update, 2014–2019," *Tech. Rep*, 2015.

[2] J. Erman, A. Gerber, M. Hajiaghayi, D. Pei, S. Sen, and O. Spatscheck, "To cache or not to cache: The 3g case," *IEEE Internet Computing*, vol. 15, no. 2, pp. 27–34, 2011.

[3] *Microsoft Smooth Streaming*, Available: https://www.iis.net/downloads/microsoft/smooth-streaming.

[4] *Adobe HTTP Dynamic Streaming*, Available: http://www.adobe.com/products/hds-dynamic-streaming.html.

[5] *Apple HTTP Live Streaming*, Available: https://developer.apple.com/streaming/.

[6] T. Stockhammer, "Dynamic adaptive streaming over http–: standards and design principles," in *Proceedings of the second annual ACM conference on Multimedia systems*. ACM, 2011, pp. 133–144.

[7] T. Wiegand, G. J. Sullivan, G. Bjontegaard, and A. Luthra, "Overview of the h. 264/avc video coding standard," *IEEE Transactions on circuits and systems for video technology*, vol. 13, no. 7, pp. 560–576, 2003.

[8] E. Thomas, M. van Deventer, T. Stockhammer, A. C. Begen, and J. Famaey, "Enhancing mpeg dash performance via server and network assistance," 2015.

[9] N. McKeown, T. Anderson, H. Balakrishnan, G. Parulkar, L. Peterson, J. Rexford, S. Shenker, and J. Turner, "Openflow: enabling innovation in campus networks," *ACM SIGCOMM Computer Communication Review*, vol. 38, no. 2, pp. 69–74, 2008.

[10] N. McKeown, "Software-defined networking," *INFOCOM keynote talk*, vol. 17, no. 2, pp. 30–32, 2009.

[11] S. Akhshabi, A. C. Begen, and C. Dovrolis, "An experimental evaluation of rate-adaptation algorithms in adaptive streaming over http," in *Proceedings of the second annual ACM conference on Multimedia systems*. ACM, 2011, pp. 157–168.

[12] T.-Y. Huang, N. Handigol, B. Heller, N. McKeown, and R. Johari, "Confused, timid, and unstable: picking a video streaming rate is hard," in *Proceedings of the 2012 ACM conference on Internet measurement conference*. ACM, 2012, pp. 225–238.

[13] J. Jiang, V. Sekar, and H. Zhang, "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*. ACM, 2012, pp. 97–108.

[14] Z. Li, X. Zhu, J. Gahm, R. Pan, H. Hu, A. C. Begen, and D. Oran, "Probe and adapt: Rate adaptation for http video streaming at scale," *IEEE Journal on Selected Areas in Communications*, vol. 32, no. 4, pp. 719–733, 2014.

[15] A. Bentaleb, A. C. Begen, and R. Zimmermann, "Sdndash: Improving qoe of http adaptive streaming using software defined networking," in *Proceedings of the 2016 ACM on Multimedia Conference*. ACM, 2016, pp. 1296–1305.

[16] A. Bentaleb, A. C. Begen, R. Zimmermann, and S. Harous, "Sdnhas: An sdn-enabled architecture to optimize qoe in http adaptive streaming," *IEEE Transactions on Multimedia*, vol. 19, no. 10, pp. 2136–2151, 2017.

[17] X. Yin, A. Jindal, V. Sekar, and B. Sinopoli, "A control-theoretic approach for dynamic adaptive video streaming over http," *ACM SIGCOMM Computer Communication Review*, vol. 45, no. 4, pp. 325–338, 2015.

[18] S. Xiang, L. Cai, and J. Pan, "Adaptive scalable video streaming in wireless networks," in *Proceedings of the 3rd multimedia systems conference*. ACM, 2012, pp. 167–172.

[19] R. Kuschnig, I. Kofler, and H. Hellwagner, "An evaluation of tcp-based rate-control algorithms for adaptive internet streaming of h. 264/svc," in *Proceedings of the first annual ACM SIGMM conference on Multimedia systems*. ACM, 2010, pp. 157–168.

[20] V. Krishnamoorthi, N. Carlsson, D. Eager, A. Mahanti, and N. Shahmehri, "Helping hand or hidden hurdle: Proxy-assisted http-based adaptive streaming performance," in *Modeling, Analysis & Simulation of Computer and Telecommunication Systems (MASCOTS), 2013 IEEE 21st International Symposium on*. IEEE, 2013, pp. 182–191.

[21] P. Georgopoulos, Y. Elkhatib, M. Broadbent, M. Mu, and N. Race, "Towards network-wide qoe fairness using openflow-assisted adaptive video streaming," in *Proceedings of the 2013 ACM SIGCOMM workshop on Future human-centric multimedia networking*. ACM, 2013, pp. 15–20.

[22] J. W. Kleinrouweler, S. Cabrero, and P. Cesar, "Delivering stable high-quality video: an sdn architecture with dash assisting network elements," in *Proceedings of the 7th International Conference on Multimedia Systems*. ACM, 2016, p. 4.

[23] H. E. Egilmez, S. Civanlar, and A. M. Tekalp, "An optimization framework for qos-enabled adaptive video streaming over openflow networks," *IEEE Transactions on Multimedia*, vol. 15, no. 3, pp. 710–715, 2013.

[24] H. E. Egilmez and A. M. Tekalp, "Distributed qos architectures for multimedia streaming over software defined networks," *IEEE Transactions on Multimedia*, vol. 16, no. 6, pp. 1597–1609, 2014.

[25] H. E. Egilmez, S. T. Dane, K. T. Bagci, and A. M. Tekalp, "Openqos: An openflow controller design for multimedia delivery with end-to-end quality of service over software-defined networks," in *Signal & Information Processing Association Annual Summit and Conference (APSIPA ASC), 2012 Asia-Pacific*. IEEE, 2012, pp. 1–8.

[26] N. Bouten, J. Famaey, S. Latré, R. Huysegems, B. De Vleeschauwer, W. Van Leekwijck, and F. De Turck, "Qoe optimization through in-network quality adaptation for http adaptive streaming," in *Network and service management (cnsm), 2012 8th international conference and 2012 workshop on systems virtualiztion management (svm)*. IEEE, 2012, pp. 336–342.

[27] R. K. Mok, X. Luo, E. W. Chan, and R. K. Chang, "Qdash: a qoe-aware dash system," in *Proceedings of the 3rd Multimedia Systems Conference*. ACM, 2012, pp. 11–22.

[28] A. El Essaili, D. Schroeder, E. Steinbach, D. Staehle, and M. Shehada, "Qoe-based traffic and resource management for adaptive http video delivery in lte," *IEEE Transactions on Circuits and Systems for Video Technology*, vol. 25, no. 6, pp. 988–1001, 2015.

[29] A. Montazerolghaem, M. H. Yaghmaee, and A. Leon-Garcia, "Opensip: Toward software-defined sip networking," *IEEE Transactions on Network and Service Management*, 2017.

[30] *Open vSwitch*, Available:http://docs.openvswitch.org/.

[31] S. Lederer, C. Müller, and C. Timmerer, "Dynamic adaptive streaming over http dataset," in *Proceedings of the 3rd Multimedia Systems Conference*. ACM, 2012, pp. 89–94.

[32] Z. Wang, L. Lu, and A. C. Bovik, "Video quality assessment based on structural distortion measurement," *Signal processing: Image communication*, vol. 19, no. 2, pp. 121–132, 2004.

[33] B. Lantz, B. Heller, and N. McKeown, "A network in a laptop: rapid prototyping for software-defined networks," in *Proceedings of the 9th ACM SIGCOMM Workshop on Hot Topics in Networks*. ACM, 2010, p. 19.

[34] *Floodlight Controller*, Available:http://www.projectfloodlight.org/.

[35] *Scootplayer*, Available: https://github.com/broadbent/scootplayer.

[36] T. Hoßfeld, L. Skorin-Kapov, P. E. Heegaard, and M. Varela, "Definition of qoe fairness in shared systems," *IEEE Communications Letters*, vol. 21, no. 1, pp. 184–187, 2017.

[37] S. Schwarzmann, T. Zinner, and O. Dobrijevic, "Quantitative comparison of application–network interaction: a case study of adaptive video streaming," *Quality and User Experience*, vol. 2, no. 1, p. 7, 2017.

[38] S. Petrangeli, T. Wauters, R. Huysegems, T. Bostoen, and F. De Turck, "Network-based dynamic prioritization of http adaptive streams to avoid video freezes," in *Integrated Network Management (IM), 2015 IFIP/IEEE International Symposium on*. IEEE, 2015, pp. 1242–1248.

[39] G. Cofano, L. De Cicco, T. Zinner, A. Nguyen-Ngoc, P. Tran-Gia, and S. Mascolo, "Design and experimental evaluation of network-assisted strategies for http adaptive streaming," in *Proceedings of the 7th International Conference on Multimedia Systems*. ACM, 2016, p. 3.