Mobile Networks and Applications
https://doi.org/10.1007/s11036-020-01543-1

An Efficient QoE-Aware HTTP Adaptive Streaming over Software Defined Networking

Pham Hong Thinh[1,2] · Nguyen Thanh Dat[1] · Pham Ngoc Nam[3] · Nguyen Huu Thanh[1] · Hien M. Nguyen[4] · Truong Thu Huong[1]

Abstract

Due to the increase in video streaming traffic over the Internet, more innovative methods are in demand for improving both Quality of Experience (QoE) of users and Quality of Service (QoS) of providers. In recent years, HTTP Adaptive Streaming (HAS) has received significant attention from both industry and academia based on its impacts on the enhancement of media streaming services. However, HAS-alone cannot guarantee a seamless viewing experience, since this highly relies on the Network Operators' infrastructure and evolving network conditions. Along with the development of future Internet infrastructure, Software-Defined Networking (SDN) has been researched and newly implemented as a promising solution in improving services of different Internet layers. In this paper, we present a novel architecture incorporating bitrate adaptation and dynamic route allocation. At the client side, adaptation logic of VBR videos streaming is built based on the MPEG-DASH standard. On the network side, an SDN controller is implemented with several routing strategies on top of the OpenFlow protocol. Our experimental results show that the proposed solution enhances at least 38% up to 185% in term of average bitrate in comparison with some existing solutions as well as achieves better viewing experience than the traditional Internet.

Keywords Dynamic routing · Adaptive streaming · SDN · DASH

1 Introduction

The last decade has witnessed a tremendous escalation of media content consumption, especially high-definition videos over the Internet. Cisco forecasts that the global Internet traffic in 2021 will equivalent to 127 times of that of the year 2005. In 2017, among the services over the Internet such as web, email, file sharing, etc., video streaming takes part in more than 74% of the global Internet traffic and will continue to rise over 81% by 2021 [1]. Those numbers figures show the necessity of developing methods in optimization of video streaming over the Internet that satisfy both efficiency for service providers and the quality for users. In that context, one technology has become the de facto standard for Internet streaming: HTTP-Based Adaptive Streaming (HAS) [2]. Using HTTP HAS is leveraging a ubiquitous and highly optimized delivery infrastructure, originally created for the web traffic, which includes, e.g., Content Delivery Networks (CDNs), caches, and proxies.

One of the enablers of the success of HAS was the open standard MPEG-DASH (Dynamic Adaptive Streaming over HTTP) [2, 3]. The fundamental principle of DASH is encoding video content into multiple versions at different discrete bitrates. Each encoded video is then fragmented into small video segments or chunks, each containing a few seconds of video. Normally, the duration of each video segment is a determined value in the range of 2s to 10s [4]. Obviously, a long segment duration has many benefits such as fewer requests, less overhead, and higher overall throughput utilization [5]. However, it results in large delay from the server to the client [6]. Moreover, the client

This paper has been accepted in part to the INISCOM conference 2019 - 5th EAI International Conference on Industrial Networks and Intelligent Systems.

✉ Truong Thu Huong
huong.truongthu@hust.edu.vn

[1] Hanoi University of Science and Technology, Hanoi, Vietnam

[2] Quy Nhon University, Binhdinh, Vietnam

[3] College of Engineering and Computer Science, VinUniversity, Hanoi, Vietnam

[4] Duy Tan University, Danang, Vietnam

Published online: 17 May 2020

could only adapt to network changes after it receives a whole video segment, causing slow adaptability and, as a result, buffer instability. A straightforward solution is to use a short segment duration, which certainly introduces an explosion in the number of requests. Request-related overheads, especially when segment duration is small, have been pointed out in some recent studies (e.g., [6, 7]). DASH enables clients to request a segment's representation actively which avoids the overhead computation at the server when numerous clients connect to the server simultaneously. Rate adaptation algorithms of HAS clients were often categorized into three categories, namely, throughput based, buffer-based and hybrid method throughput-buffer based.

From another aspect, Software-Defined Networking [8] is a new network architecture, that centralizes network intelligence in one network component by disassociating the forwarding process of network packets (data plane) from the routing process (control plane). In SDN, the common logical architecture in all switches, routers, and other network devices are managed by an SDN controller that allows networks policies to be dynamically designed to support every single specific application. The design and application of the SDN architecture can be found in various researches and network contexts lately such as [9–14].

In the network terminology, the user's experience is called Quality of Experience – QoE. In fact, this user's perceived experience is significantly affected by the video quality level that the client is going to select. Obviously, if the quality level suddenly drops, users will suffer a worse playback and the gradual falling is better for their experience. In addition, choosing too high-quality level can cause video stalling which can be measured by the frequency of stalling event and stalling duration which are important influences to users' experience.

In this study, we develop new methods to unite the advantages of dynamic adaptive streaming over HTTP and Software-Defined Networking. At the client side, the proposed adaptation algorithm based on DASH that frequently requires quality levels of video segments (a bitrate level of the video segment, a high-quality level means a high video quality) up or down depending on the bandwidth and client's buffer status. Within the transportation network, in order to improve the bandwidth received at the client, two routing policies are proposed, called periodically routing and on-demand routing. Experimental results prove that our approach out-performs the existing state-of-the-art streaming solutions.

The remainder of this paper is organized as follows. Section 2 provides related work on several existing adaptation methods and bandwidth allocation schemes. Section 3 introduces our bitrate adaptation algorithm and SDN-based dynamic routing solutions. The experiment setup and performance evaluation of the proposed solution are presented in Section 4. Conclusion and possible future extensions are presented in the last section.

2 Related work

The DASH standard is designed to cope with highly varying delivery conditions. Over the past few years, many bitrate adaptation algorithms have been introduced in order to improve user's Quality of Experience (QoE). Their difference is mainly the required input information, ranging from network characteristics to application-layer parameters such as the playback buffer or the download speed. In DASH, all algorithms decide the bitrate of the next download segment based on the throughput variation or buffer level at the client side. These algorithms can be roughly divided into three main types such as the throughput-based group [15–18] the buffer-based group [19–22] and the mixed type of rate adaptation algorithms [23–26].

In throughput-based methods, bitrate is decided based on the estimated throughput without considering buffer. These schemes mainly aim at dynamically adapting a video bitrate to an available bandwidth, which usually leads to low bandwidth utilization and cannot reach the maximum quality allowed by the available bandwidth. This is because, in such schemes, video bitrates higher than available bandwidth are never allowed to be selected to avoid playback interruptions. The key differences between these methods are the ways to estimate and use the throughput. As discussed in [15, 18], authors proposed a rate adaptation algorithm based on the estimated throughput (called aggressive method) where the last segment throughput is simply used as the estimated throughput. It is currently the most responsive method to capture the dynamic changes in throughput. In the aggressive method, the bitrate is decided as the highest bitrate that is lower than the estimated throughput. This algorithm is sensitive with highly variable throughput such as short-term bitrate peaks. In these situations, the client will request the high-quality level and cannot be able to completely download a segment before the throughput suddenly falling down. And the rest of the heavy segment will be downloaded in a poor bandwidth condition, eventually, the buffer will run dry, playback freezes. In some other researches we can find proposed models of encoding rate allocation such as [27, 28] tries to optimize QoE, and minimizing the bandwidth resource. QoE in these work is considered as high hit rate to access the videos, continuity and smooth streaming with low quality fluctuation. Or work [29, 30] proposed an encoding allocation model provides high playback quality of received videos. However these approaches are designed for video streaming over either 5G communications or wireless communication. The authors in

[31, 32] proposed to optimize video streaming in terms of energy and resource saving. However, the video streaming is not optimized from the QoE point of view.

Buffer-based schemes basically employ buffer thresholds to decide the changes of bitrate. Compared to the throughput-based methods, buffer-based methods provide smoother video bitrate curves in on-demand streaming. Usually, during a certain range of buffer level, a client will try to maintain the current bitrate, resulting in a stable bitrate curve and a rather unstable buffer level. However, when bandwidth is drastically reduced, the buffer-based methods may cause a sudden change of bitrate, still. This is mainly because there is a trade-off between the stability of buffer occupancy and the smoothness of video bitrate due to the time-varying bandwidth. In [21, 22], Huang et al. proposed a class of buffer-based bitrate adaptation algorithm for HTTP video streaming, called BBA, that is based only on the current playback buffer occupancy as bitrate is selected heuristically without throughput estimation. The objective of BBA is to maximize the average video quality by selecting the available highest bitrate level that the network can support and avoid stalling events. However, this scheme exhibits many limitations such as decreasing the QoE in the case of long-term bandwidth variations and slow bandwidth fluctuation detection.

There are some ABR (Adaptive Bitrate) algorithms in the literature that consider also the path bandwidth combining with the current buffer occupancy to select the most suitable video version for the next segment. In [23, 24], the authors proposed a rate adaptation algorithm for VBR video which based on buffer thresholds to request the next video quality level. The study in [25] proposed throughput-buffer based algorithm. The algorithm depends on perceived bandwidth, the video player buffer filling level, and the buffer target to decide the segment's version. The authors in [26] proposed a segment-aware rate adaptation (SARA) algorithm that considers the segment size variation in addition to the estimated path bandwidth and the current buffer occupancy to accurately predict the time required to download the next segment. This ensures that the best possible representation of the video is downloaded while avoiding video buffer starvation.

Overall, current adaptation algorithms for HTTP Adaptive Streaming adjust the quality version of video segments to achieve the highest bandwidth utilization, smooth playback as well as avoid buffer underflow or overflow. However, the algorithms almost focus on improving the adaptation policy at the client side without considering the available resources in the networks.

There are some studies proposed in the literature for HAS over SDN. X. Jin et al. [33] presents a line routing mechanism for transferring data among switches by a controller. They also define a rule for determining a routing line for respective data flows depending on topological condition or changing network. The DASH client only simply requests for content with appropriate bitrate depending on the network conditions. In [34], the authors proposed an adaptive HTTP video streaming framework utilizing the flow route selection capability of SDN networks. In this method the SDN controller reroutes DASH flows in each segment period after the client has completely downloaded. This leads to the controller being overloaded when the great number of clients accesses at the same time or when network load increases rapidly. In [35], the authors proposed an SDN architecture to monitor network conditions of streaming flow in real time and dynamically change routing paths using multi-protocol label switching to provide reliable video watching experience. In SDNHAS [36], Bentaleb et al. rely on SDN-based management and resource allocation architecture with the goal to estimates optimal QoE policies for groups of users and requests a bandwidth constraint slice allocation, while providing encoding recommendations to HAS players. However, these studies only present a general adaptation mechanism. In work [37] an energy-efficient routing was considered in which energy consumption for transmitting and receiving data are taken as a criterion for routing. But the routing scheme was not considered in conjuction with bit rate adaptation at the client side.

3 Problem formulation

In this section, we propose an HTTP adaptive streaming solution included an adaptation algorithm at client and network routing policies for video contents over Software-Defined Networking platform.

3.1 Architecture for HTTP streaming over SDN

The proposed rate adaptation architecture for video streaming over SDN is shown in Fig. 1. At the server side, the MPEG-DASH introduced the Media Presentation Description (MPD) as the video manifest file. The MPD is an XML file that represents the different qualities of the media content and the individual segments of each quality with HTTP Uniform Resource Locators (URLs). The *Content Annotation* module provides metadata about the content, ranging from semantic level to the physical level. The *Media Preparation* module provides tools for media adaptation/transcoding, packetization, encapsulation, etc., so that the media could be efficiently delivered to the client. The metadata provided by *Content Annotation* module will be requested by the client.

At the client side, *MPD Analyzer* module parses the MPD file to obtain the segment information as URLs, bitrate and

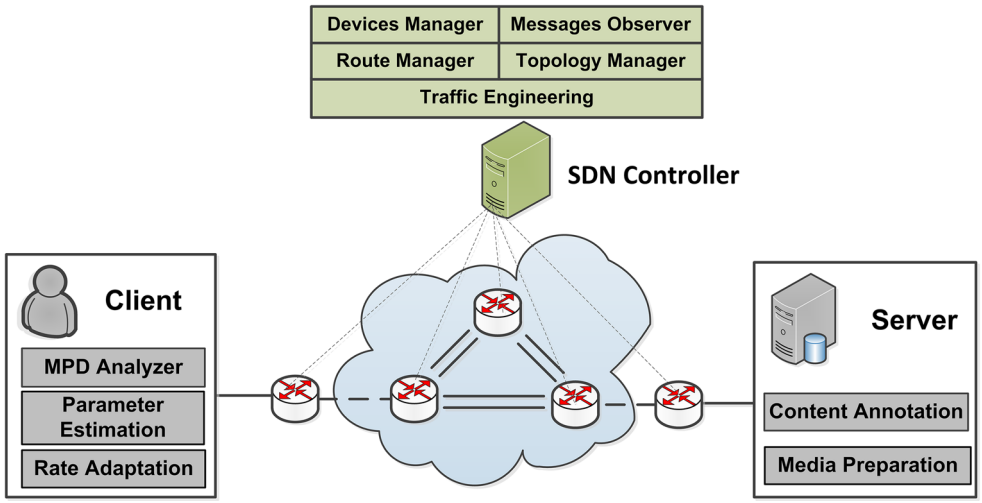

Fig. 1 Proposed architecture for HTTP adaptive video streaming over SDN

segment sizes. The *Parameter Estimation* module estimates current buffer occupancy and status of network before requesting the individual segments that fit best for its requirements. The suitable representation for the video segment is determined by *Rate Adaptation* module.

In the network, the SDN controller's main functions are delivered by several modules as follows:

- *Topology Manager*: This module is responsible for discovering the network topology i.e. routers, switches, and links by sending check-up messages on a regular basis.
- *Devices Manager*: This module supervises hosts in the network. Each host is considered as an attachment to the forwarder with which it has a direct connection.
- *Traffic Engineering*: This module collects statistical data from forwarders to determine the packet forwarding performance, which is used to help the route calculation.
- *Route Manager*: This module cooperates with Topology Manager and Traffic Manager, runs routing algorithms to obtain possible network routes between hosts as well as routes that meet some requirements. It is also responsible for installing flow rules onto routers or switches.
- *Messages Observer*: This module handles messages from application services.

When the client requests a video streaming service, it will send a connection to the nearest Openflow switch. This switch checks its flow-tables and finds out that this is a new service request as no flow-table has been configured for it. Then, the Openflow switch encodes the request's information in a *PacketIn* message and sends it to the SDN controller. Based on functional modules above, the SDN controller parses the message, checks the current network status, and calculates the video streaming scheme, which includes the video server and streaming path, for the client. Then, the SDN controller encodes the information of the streaming path in *FlowMod* messages, distributes them to the related switches to install the best path, and informs a new path from the client to the server. When a new path is set up, between the DASH client and the DASH server where the metadata/media will be delivered by a sequence of HTTP request-response transactions.

Overall, HTTP video streaming is carried out over an Openflow/SDN transportation network where Openflow switches are controlled by the SDN controller. In our design, this controller can reinforce routing policies based on network conditions, while at the client side, a bitrate adaptation algorithm can be implemented in order to acquire a seamless playback as well as a best possible video quality.

3.2 Adaptation problems and our proposed algorithm - MUNTH

Many adaptation algorithms have been proposed to attain a good experience in poor bandwidth conditions (highly

Table 1 Symbols using in the paper

Symbol	Description
i	Segment index
j	Version index
Q	The total number of available quality levels
N	Number of segments
B_i	The buffer level at segment i
I_i	The representation's index of segment i
D_i	The download rate of segment i
R_j	The bitrate of version j
B^e_{i+1}	The estimate buffer for segment i+1
B^{Th}	The buffer threshold
D^{Th}	The bitrate threshold
T_i	The measured throughput at segment i
T^e_{i+1}	The estimate throughput for segment i+1
RTT	Round trip time
SD	Segment duration

variable throughput or low bandwidth etc...). With a proper method, a client can avoid video freezing caused by sudden severe bandwidth drop and achieve an acceptable video's quality. Before presenting the proposal, three adaptation algorithms corresponding to the three well-known mechanisms as mentioned before will be introduced first. Table 1 showsx the symbols used in this study.

The authors in [15, 18] proposed the Aggressive algorithm based on throughput without considering the buffer's condition. The next throughput T^e_{i+1} is assigned equal to the current throughput and the highest possible value of segment bitrate R_j is computed by the estimated throughput and a safety margin μ as in Eq. 1 with μ is range from 0 to 0.5.

$$I_{i+1} = \arg\max_{\mathbf{j}} \left\{ R_j \mid R_j \leq (1-\mu) \times T^e_{i+1} \right\} \tag{1}$$

BBA is a very well-known buffer-based adaptation algorithm. According to BBA [21, 22], the average segment size of each corresponding bitrate is mapped to an instantaneous buffer level, in a linear manner with two fixed points for the lowest and highest bit-rate. BBA is too conservative during startup. The network can sustain a much higher video rate, but the algorithm is just not aware of it yet. There is a trade-off between buffer occupancy and video quality in BBA.

The Segment Aware Rate Adaptation - SARA [26] estimated next weighted Harmonic Mean throughput based on all measured throughputs in the past as Eq. 2, where w_i and d_i are the volume and the download rate of segment number i, respectively. SARA selects the most suitable representation for the next segment to be downloaded based on H_n the buffer occupancy at any given time, B_{curr}. The rate adaptation is done in four stages including fast start, addictive increase, aggressive switching, delayed download corresponding to B_{curr} value.

$$H_n = \frac{\sum_{i=1}^{n} w_i}{\sum_{i=1}^{n} \frac{w_i}{d_i}} \tag{2}$$

3.2.1 Our proposed adaptation algorithm – MUNTH (iMpede sUspend and attaiN poTential patH)

In our solution, next throughput is estimated based on two previous measured throughput at client as described in Eq. 3 [9], where γ is a dynamic constant range from [0,1]. This enables a client to adapt well with the highly bandwidth fluctuations especially with highly variable bandwidth which is the main cause of video freezing. The γ is usually set to be 0.5, when γ = 1.0, and the estimation is exactly the same as aggressive method.

$$T^e_{i+1} = \gamma \times T_i + (1-\gamma) \times T_{i-1} \tag{3}$$

To avoid video freezing, we propose a new quality selection mechanism based on the estimate buffer level. Equation 4 is used to estimate the next buffer level if client selects quality level j for segment number i where SD is segment duration. Our objective is to choose a suitable quality level to keep the buffer greater than a threshold B^{Th} to prevent stalling events. The details of our method are shown in Algorithm 1.

$$B^e_{i+1} = B_i + SD - RTT - \frac{SD \times R_i}{T^e_{i+1}} \tag{4}$$

Algorithm 1 Bitrate adaptation algorithm - MUNTH.

Input: T_i, R_n, B_i, B^{Th}, D_i, RTT, D^{Th}, SD
Output: I_{i+1}

```
T^e_{i+1} ← γ × T_i + (1 − γ) × T_{i−1};          // Estimate throughput.
I_{i+1} ← 0
if D_i ≤ D^{Th} then
    Request for a new path;
else
    for j ← Q − 1 to 0 do
        B^e_{i+1} ← B_i + SD − RTT − SD×R_j / T^e_{i+1};   // Estimate buffer level.
        if B^e_{i+1} ≥ B^{Th} then
            I_{i+1} = j;
        end
    end
end
```

In the proposed algorithm, when the download rate D_i is smaller than D^{Th} = 1000kbps, client is going to send a message to the network controller to request an optimal path which is sufficiency to transport video data. The detail policies at network controller will be discussed in the next subsection.

3.3 Routing policies

With the OpenFlow/SDN-based architecture as shown in Fig. 1 which can control and manage all types of data flows in the control layer, the SDN platform provides ability to flexibly perform any routing rules in the network. Furthermore, the centralized scheme of SDN has the ability to entirely monitor the network topology and routing status, and timely modify the path selection according to the changes of network states.

In SDN networks, when a stream established, the first request message from client will be forwarded to the Controller, Route Manager determines the best-effort path from client to server and update flow entries on forwarding devices. Traffic Manager is also active to monitor current network resource and it may trigger the routing module again to find a new route according as the traffic policy

3.3.1 Periodical routing

We proposed a periodical routing mechanism to select the optimal path every T seconds based on the stability and availability of each path. Every path from a client to the server will be represented by $Repre^p$ as shown in Eq. 5.

$$Repre^p = (1 - \omega^p) \times BW^p_{inst} \tag{5}$$

$Repre^p$ is calculated from the stability w^p and the availability BW^p_{inst} which are:

- BW^p_{inst}: The instant bandwidth of each path measured by the SDN controller, p represents for path number p in n available paths (indexed from 1 to n) from a client to the server. If a path includes multiple links, BW^p_{inst} is assigned to the bandwidth of the bottleneck link.
- w^p: The stability of a path p, calculated by the Eq. 6. With BW^p_i is the measured bandwidth on path p at $i \times T$ seconds before, numerator is the standard deviation of m previous bandwidth values measured in every T seconds of path p and the denominator is the sum of standard deviation of n available path. w^p is in range [0,1] and the higher w^p is, the less stable path p is.

$$\omega^p = \frac{\sqrt{\frac{\sum_{i=1}^{m}(BW_i^p - \overline{BW^p})^2}{m}}}{\sum_{p=0}^{n-1}\sqrt{\frac{\sum_{i=1}^{m}(BW_i^p - \overline{BW^p})^2}{m}}} \tag{6}$$

The Controller selects the most stable and available path which has the highest $Repre^p_{inst}$ as shown in Eq. 7 where $Indx$ is the selected path index.

$$Indx = indexOf(\max_{p=\overline{0..n-1}}\{Repre^p\}) \tag{7}$$

If there are multiple paths from client to server, periodical routing mechanism consumes a heavy computation as well as the memory on the controller to select the optimal path. This can avoid by increase the Routing Period T but it will affect the accuracy of paths' stability w^p when the bandwidth value far away from the present is used.

3.3.2 Adaptive routing - MUNTH

We propose an active mechanism from client namely MUNTH to request for the optimal path when the current bandwidth is not satisfying their demand. A client can send a 'reroute' message to the controller when download rate is lower than 1000kbps and the controller then selects the optimal path having highest BW^p_{inst} to serve the stream as shown in Eq. 8. the controller only has to seek a new path when poor network conditions occur otherwise the path will be kept during the stream session. This mechanism has two advantages: firstly, it reduces computation at the controller compared to the periodical routing scheme. Secondly, a client knows best about its perceived network condition as well as its ability, therefore, giving the client an ability to control the adaptive rates is meaningful.

$$Indx = indexOf(\max_{p=\overline{0..n-1}}\{BW^p\}) \tag{8}$$

4 Performance evaluation

4.1 Experiment setting

In this section, we present the experimental setting as well as evaluate the performance of our proposed scheme. The setup video streaming system includes three parts: DASH client, SDN network and Video server. SDN Network topology is created by Mininet [38] including 5 switches controlled by Floodlight [39] controller. Client and server are connected to the network via switch s1 and s2 respectively as in Fig. 2. We can see from the topology that there are 4 available paths from server to client:

1. s2 → s3 → s1
2. s2 → s3 → s4 → s1
3. s2 → s5 → s3 → s1
4. s2 → s5 → s4 → s1

To simulate the bandwidth fluctuation, Traffic Control [40] are used to assign desired bandwidth on every link in topology. Bandwidth on every path and the optimal path are shown in Fig. 3.

The client is installed by libdash library [41] on linux to serve the DASH standards. Client's media player is implemented by Qtsampleplayer [41] with buffer size of 50s. The emulated server stores a VBR video clip named "Elephants Dream" [42] with a length of 10 min 52s, 2 seconds segment, 24fps and Full-HD resolution

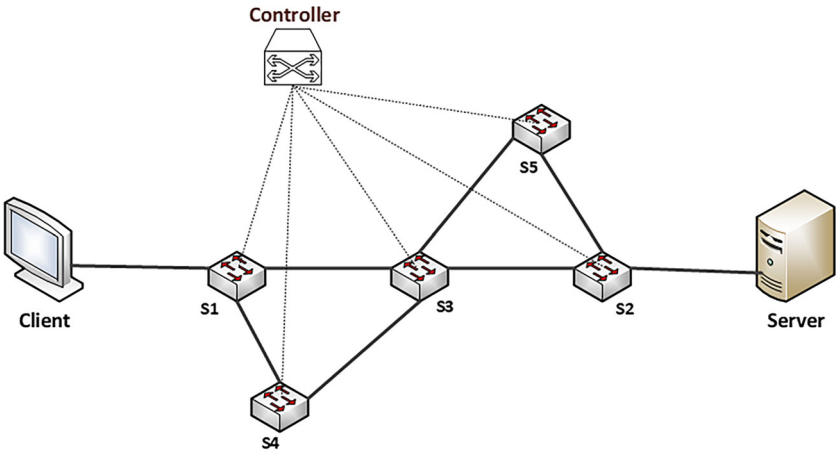

Fig. 2 Network topology used in experiments

(1920x1080). Every segment is encoded to 12 different versions corresponding to 12 QP values (13 to 46) of the H.264 standard. The parameters set up in our experiment can be summarized in Table. 2. As seen in Table. 2, buffer is considered in second due to the fact that in video streaming systems, segments are downloaded and stored in buffer. The duration of buffer will directly affect the playback experience, not the size of the buffer. That is why within the scope of our research, buffer is measured in second.

The detailed information about version level and average bitrate value are shown in Table 3.

In this paper, we test our solution in two scenarios and six experiments as follows:

***Scenario 1*:** Buffer threshold optimization

At this scenario multiple B^{Th} values are used to evaluate the effect of B^{Th} to proposal's performance.

- **Experiment 1:** The client runs the MUNTH adaptation algorithm with B^{Th} of 10s, 15s, 20s, 25s with the adaptive routing policy at the controller ($m = 5$).

***Scenario 2*:** Performance comparison

To evaluate the performance of MUNTH, we investigate the related existing schemes by setting up as follows:

- **Experiment 2:** Named AGG_DF, the client runs the Aggressive algorithm, the SDN controller is active with the default routing policy (e.g. Dijkstra algorithm).
- **Experiment 3:** Named SARA, the client runs the SARA algorithm, the SDN controller have the same configuration as Experiment 2.
- **Experiment 4:** Named BBA, the client runs the BBA algorithm (BBA parameter), the SDN controller have the same configuration as Experiment 2.
- **Experiment 5:** Named AGG_RR, the client runs Aggressive algorithm, the SDN controller now uses periodical routing policy.
- **Experiment 6:** Named MUNTH, the client runs MUNTH algorithm with the optimal B^{Th} selected from Experiment 1.

Safety margin μ is set equal to 0.1 in all experiments.

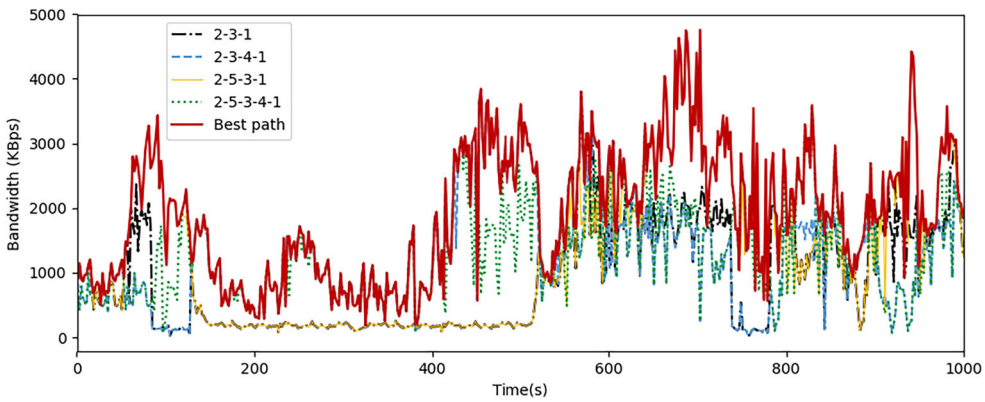

Fig. 3 Simulation bandwidth on paths and the highest bandwidth path

Table 2 Parameters used in our experiment

Parameters	Value
Q	12
N	327
B_i	0s-50s
B^{Th}	10s, 15s, 20s, 25s
D^{Th}	1000Kbps
RTT	50ms
SD	2s

4.2 QoE model

In this study, two groups of experiments were designed to evaluate the performance of the adaptive streaming service's QoE (Quality of Experience). The QoE can only be estimated, either by subjective test or using an objective model of the users' perception. The definition of QoE started at the beginning by measuring truly-subjective feeling of end users. For example, researches that use the subjective QoE (i.e MOS - Mean Opinion Square) for video streaming can be found in work [43] and [44]. Table 4 illustrates QoE ranging from 1 to 5 to represent overall quality perceived by users.

QoE perceived by the end user is influenced by the characteristics recorded from experiments. The most important characteristics are bitrate distribution, average bitrate, number of bit rate changes, starting bitrate, stall time (total stall time, average stall duration and stall distribution) and stagnant number [45]. The users' experience is strongly affected by the received quality level, the higher level obviously offers the better experience and users are dissatisfied with the dramatic drop of quality. Especially the QoE can be degraded to a bad level only by one stall which

Table 3 The version index, QP value and Average Bitrate of each quality level

Version index	QP	Average Bitrate(kbps)
0	46	354
1	43	472
2	40	638
3	37	882
4	34	1234
5	31	1779
6	28	2588
7	25	3824
8	22	5613
9	19	8028
10	16	11156
11	13	15227

Table 4 QoE values corresponding to the user's experience

Quality of experience	Quality
between 4 and 5	Excellent
between 3 and 4	Good
between 2 and 3	Fair
between 1 and 2	Poor
between 0 and 1	Bad

usually happened in the fluctuation bandwidth conditions. To evaluate the performance of streaming methods, we assess QoE of an end user under the form of a prediction for the MOS (mean opinion score) of an audience. In [46], Claeys et al. defined the QoE (i.e. MOS) of a HAS video to be dependent on three aspects of the video playout including the average segment quality, quality switches and video freezes. The parameters of this quality level model were tuned based on the results of a small subjective test.

$$\lambda = \frac{7}{8} \times \max\left(\frac{\ln(FF)}{6} + 1, 0\right) + \frac{1}{8} \times \left(\frac{\min(AFT, 15)}{15}\right) \tag{9}$$

The authors in [46] take into account the impact of video freezes on the QoE prediction. The influence of video freezes λ as shown in Eq. 9 which depends on the frequency of freeze (FF) and the average time of freezes (AFT).

The quality switches factor β is shown in Eq. 10.

$$\beta = \frac{N_{sw} \times AVG_{depth}}{N * (Q - 1)} \tag{10}$$

where N_{sw} and AVG_{depth} are the number of quality switches and average switch depth respectively.

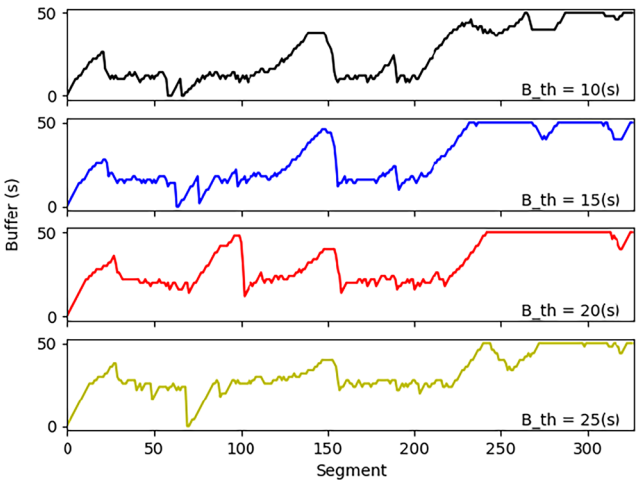

Fig. 4 Buffer level of threshold experiments

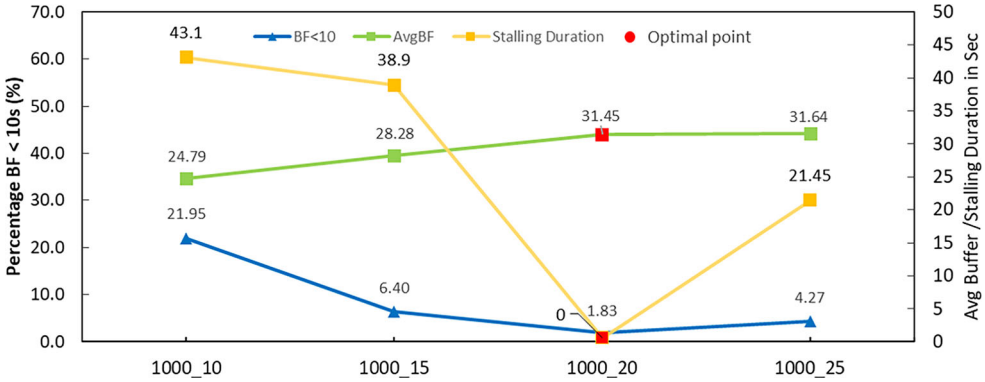

Fig. 5 Buffer statistics of the MUNTH method

The estimated MOS (or QoE) for the playout of a HAS video was calculated using Eq. 11 proposed by [46].

$$QoE = M_{pred} = 4.85 \times \frac{\alpha}{Q} - 1.57 \times \beta - 4.95 \times \lambda + 0.5 \quad (11)$$

with Q is quality levels; and α is the average quality level requested. The theoretical range of this metric is [3:76; 5:35]. However, the practical range with realistic simulations of this equation is [0.0, 5.35].

Equation 11 shows that to enhance QoE, the clients have to avoid quality switching and video freezing in both frequency as well as the amplitude aspect while keeping a stable quality level.

4.3 B^{Th} optimization for MUNTH

Figure 4 show the occupied buffer for different B^{Th} value during stream session (327 segment). As we can see, the higher B^{Th} keeps the occupied buffer in the higher level which can avoid stalling event. When B^{Th} =10s, 15s video still freezes 2 time. However, with higher B^{Th} = 25s, the video freezes only 1 time especially when B^{Th} =20s, MUNTH achieve a smooth playback.

In Fig. 5, the percentage of buffer smaller than 10s, average buffer, and total stalling duration are shown for different B^{Th}. As we can see, with B^{Th} =10s the stalling duration is highest at 43.1s. It also has the highest rate of occupied buffer lower than 10s and lowest average buffer. In contrast, experiment with B^{Th} =20s shows the best results. Stalling duration is zero, the percentage of buffer $\leq$ 10s is only 1.83% while keeping an average buffer at a comparable level than B^{Th} =25s.

However, a higher B^{Th} can harm the video quality because of client concentrate on avoiding stalling events more than video representation. Quality metrics from experiment results are shown in Table 5. As we can see, when B^{Th} =25 both average bitrate and percentage of bitrate $\geq$ 8000kbps is smaller than the rest.

In conclusion, the experiment results show that our MUNTH algorithm with B^{Th} =20 enables a client to occupy enough buffer to avoid freezing event while achieving a comparable video quality compared with the others. The optimal value for B^{Th} was chosen is 20s.

Table 5 MUNTH Performance with different B^{Th} values

Criteria	B^{Th}=10	B^{Th}=15	B^{Th}=20	B^{Th}=25
Stalling event	6	2	0	3
Stalling duration	43.1	38.9	0	21.45
Percentage Buffer $\leq$ 10s(%)	21.95	6.4	1.83	4.27
Avg Birate	10613.64	10496.54	10764.62	9575.02
Bitrate $\geq$ 8000 kbps	46.34	44.21	46.34	40.55
Switching down version	42	33	39	44

4.4 Performance comparison

In this part, the MUNTH algorithm with optimized B^{Th}=20 will be compared with the other algorithms as listed in *Scenario* 2.

Figure 6 shows the occupied buffer for five different experiments respectively. Because Aggressive algorithm

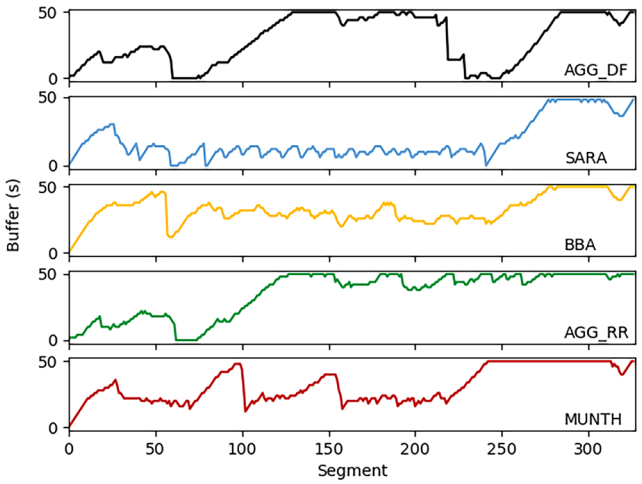

Fig. 6 Buffer level of all methods

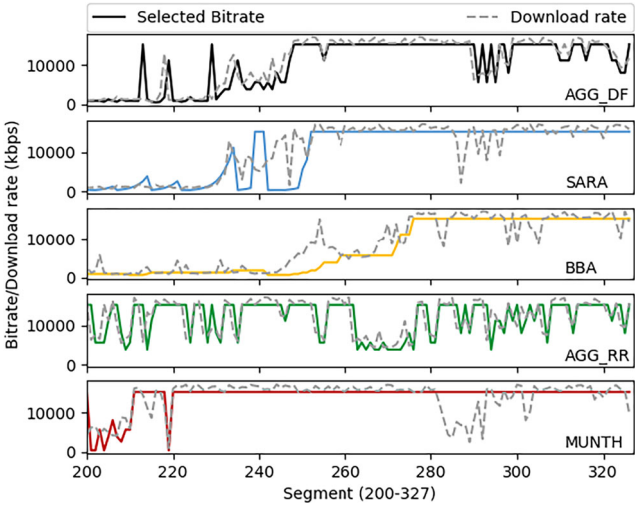

Fig. 7 Measured throughput and selected bitrate of all five methods

does not have mechanisms to avoid bandwidth fluctuation therefore in AGG_DF experiment, stalling events occur regularly in the period 50-100 and 200-250. By safely selecting bitrate version when buffer level in fast start stage as well as using weighted harmonic download rate H_n to avoid falling into buffer trap, the SARA experiment's result shows less stalling event than aggressive. In BBA experiment, buffer level is always above the dead−line because it picks the video rate as a function of current buffer occupancy and only chose the minimum rate when buffer level is in the reservoir. This enable buffer starts to grow before it runs dry. The periodical routing mechanism enables AGG_RR to delivery segments on a stable path and a client can achieve smoother streaming compared to AGG_DF . The result for AGG_RR shows that in only the duration from segment 50 to 100 stalling events occur. MUNTH successfully avoids stalling events (0 times) by smooth throughput estimation and selection of suitable video rate to preserve buffer on an optimal path.

Figure 7 shows the download rate and selected birate of five methods, with the color line (solid line) and grey line (dashed line) are for the selected bitrate and download rate respectively. The first thing to notice from Fig. 7 is MUNTH achieve a stable quality level compared to the 4 counterparts. From segment 220 to 327, MUNTH only selects one quality level while from segment 280-327 saw a high bandwidth fluctuation. BBA is a buffer-based method, the bitrate selection is not directly depending on network conditions, but the higher bandwidth enables BBA to request a higher bitrate level because of the increase in buffer occupancy. Two aggressive methods show a significant fluctuation in the quality level as the bandwidth because of its throughput-based mechanism. With mixed throughput and buffer mechanism SARA, it can adapt well with the bandwidth fluctuation to achieve a stable bitrate.

From network perspective, Fig. 8 show Cumulative Distribution Function (CDF) of download rate (Fig. 8a) and bitrate (Fig. 8b) during playing time. The first thing to notice from the bitrate CDF figure is that the percentage of segments downloaded at the high bitrate of MUNTH is greater than the rest. Specifically, around 40% of segments have bitrate greater than 10000 kbps. The CDF value for AGG_DF, AGG_RR and SARA is lower about 20-25% while the figure for BBA is lowest at 10%. We can see from Fig. 8a that MUNTH and AGG_RR offer a higher download rate compared to fixed path algorithms. The proportion of segments downloaded with download rate $\leq$ 5000kbps of MUNTH and AGG_RR are around 25% and 40% respectively. While the figure for AGG_DF, SARA, and BBA with the fixed path is much higher at around 60%. The active routing policy of MUNTH enables about 60% of segments are downloaded with download rate $\geq$ 10000kbps. However, the CDF for the other algorithms is only from 20-35%.

Table 6 shows the precise numerical metrics accumulated during our test. AGG_DF uses the aggressive algorithm and fixed path. It requests the highest bitrate satisfied the throughput constraint without carrying about the current buffer level. From the table, we can see that AGG_DF has an average quality level compared to the other methods while

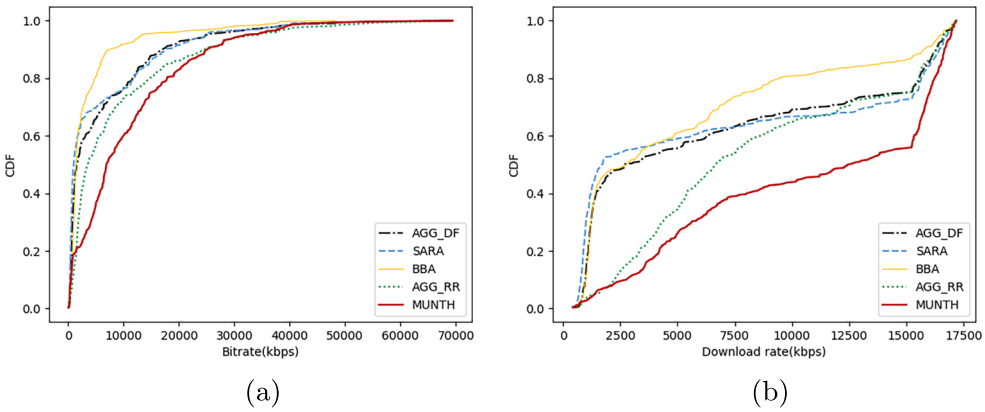

Fig. 8 Cumulative Distribution Function of **(a)** Bitrate and **(b)** Download rate

Table 6 Quality statistics of the solutions

Criteria	AGG_DF	SARA	BBA	AGG_RR	MUNTH
Stalling duration	125.11	75.32	0	81.3	0
Number of stalling events	26	8	0	13	0
Average buffer (s)	29.27	17.61	32.55	35.24	31.50
Percentage buffer $\leq$ 10s (%)	20.43	42.07	1.83	13.72	1.83
Average bitrate (kbps)	6331.33	5953.73	3754.68	8800.26	10734.18
Percentage bitrate $\geq$ 8000 kbps (%)	25.91	25.91	9.76	31.71	46.34
Number of version switch-downs	79	21	25	105	39
Switch path	-	-	-	23	2
QoE	0.84	1.32	3.19	1.98	4.32

having the highest freeze frequency and freeze duration. SARA is the mixed throughput - buffer algorithm which can reduce stalling events compared to AGG_DF but its $Average\ Bitrate$ is slightly lower. BBA successfully avoids stalling events. However, the $Average\ Bitrate$ (in kbps) of BBA is lowest at only 3754.68 kbps. By using the periodical rerouting policy, AGG_RR acquires a higher bitrate than AGG_DF and can avoid freezing by choosing the most stable path every T second. MUNTH shows a superior result compared to the other methods. Firstly, buffer statistics show that MUNTH can avoid stalling events while keeping a high buffer occupancy ($Average\ Buffer$ =31.50s). Secondly, by the adaptive routing policy, MUNTH achieves the highest $Average\ Bitrate$ at 10734.18 kbps - an enhancement about 69.5%, 80.3%, 185.9%, 21.79% compared to AGG_DF, SARA, BBA and AGG_RR respectively. Another quality metric is the number of switching-down version event, users are sensitive with the changing in the video quality, especially with sudden drop of bitrate versions. As we can see, MUNTH obtains the comparable switch-down version with BBA and SARA. Whereas achieving such better performance, MUNTH consumes much less computation at the controller compared with AGG_RR by a politic switching path mechanism.

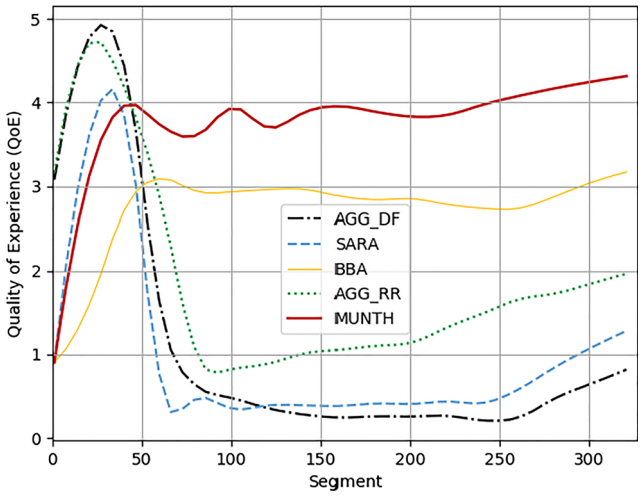

Fig. 9 Predicted MOS score

According to the formula of QoE described in Eq. 11, we calculated the final QoE scores resulted from MUNTH in comparison with 4 other solutions such as AGG_DF, SARA, BBA, AGG_RR (Table 6) in which MUNTH demonstrates to achieve highest QoE score (i.e. MOS) among the competitors. Figure 9 shows the predicted QoE during playback. The QoE of the solutions gradually increases to a good quality until triggering quality switches and video freeze caused by high bandwidth fluctuation from segment 60. In the rest of video, with dynamic path selection policy and suitable rate adaptation algorithm, MUNTH not only keeps the stable and highest QoE compared to others solutions but also achieves the excellent experience (from segment 251 to 327). BBA with a gradually quality switch is able to avoid video freeze then achieve a better performance compare to AGG_DF, SARA, and AGG_RR.

5 Conclusion

In this article we proposed a combined solution both from the client and network perspective to enhance users' experience while using HTTP Adaptive Streaming applications over SDN network. From the client side, our proposed adaptation algorithm - MUNTH uses smooth throughput estimation and buffer occupancy estimation mechanism to select a segment representation which helps the client to deal with bandwidth fluctuation therefore able to request a suitable bitrate version without harming buffer level leading to stalling event. From the network side, we proposed two routing policies, periodical routing and adaptive routing – MUNTH. Two metrics named stability w^p and availability BW^p_{inst} of a path are used to select the optimal path from the client to the sever in periodical routing manner. MUNTH routing policy is a client active scheme, where the client can actively request a new path that best satisfies its requirement. The experiment results show

that our proposal provided a better quality of experience compared to the predecessors.

For future work, we will solve a problem where topology is more complicated and multiple clients connect to multiple servers. Therefore, the problem is not only about rate adaptation and path selection but also about fairness, stability and resource utilization among clients.

Acknowledgements This work was supported by the University project grant, ID [T2018-PC-065]. The grant is funded by Hanoi University of Science and Technology.

References

1. Networking Index CV (2016) Forecast and methodology, 2016-2021, white paper. San Jose, CA, USA 1. https://www.cisco.com/c/en/us/solutions/collateral/service-provider/visual-networking-index-vni/complete-white-paper-c11-481360.html
2. Stockhammer T (2011) Dynamic adaptive streaming over http–: standards and design principles, pp 133–144
3. Sodagar I (2011) The mpeg-dash standard for multimedia streaming over the internet. IEEE MultiMedia 18(4):62–67
4. Thang TC, Le HT, Nguyen HX, Pham AT, Kang JW, Ro YM (2013) Adaptive video streaming over http with dynamic resource estimation. J Commun Netw IEEE 15(6):635–644
5. Nguyen DV, Le HT, Nam PN, Pham AT, Thang TC (2016) Adaptation method for video streaming over http/2. IEICE Communications Express 5(3):69–73
6. Wei S, Swaminathan V (2014) Low latency live video streaming over http 2.0. Proceedings of Network and Operating System Support on Digital Audio and Video Workshop, pp 37–42
7. Wei S, Swaminathan V (2014) Cost effective video streaming using server push over http 2.0. In: 2014 IEEE 16th international workshop on multimedia signal processing (MMSP) pp 1–5. IEEE
8. Foundation ON (2019) Open networking foundation. https://www.opennetworking.org/. Accessed 9 June 2019
9. Secinti G, Canberk B, Duong TQ, Shu L (2017) Software defined architecture for vanet: a testbed implementation with wireless access management. IEEE Commun Mag 55(7):135–141
10. Ozcevik ME, Canberk B, Duong TQ (2017) End to end delay modeling of heterogeneous traffic flows in software defined 5g networks. Elsevier Ad Hoc Networks 60:26–39
11. Van Tuyen D, Huong TT, Thanh NH, et al. (2018) Sdn-based syn proxy—a solution to enhance performance of attack mitigation under tcp syn flood. The Computer Journal 62(4):518–534
12. Nam TM, Phong PH, et al. (2018) Self-organizing map-based approaches in ddos flooding detection using sdn. In: Proc. of 2018 international conference on information networking (ICOIN), Chiang Mai, Thailand
13. Nam PN, Thanh NH, et al. (2015) A new power profiling method and power scaling mechanism for energy-aware netfpga gigabit router. Comput Netw 78:4–25
14. Thanh NH, Nam PN, et al. (2012) Enabling experiments for energy-efficient data center networks on openflow-based platform. In: Proc. of 2012 fourth international conference on communications and electronics (ICCE), Hue, Vietnam
15. Romero LR (2011) A dynamic adaptive http streaming video service for google android. Master's Degree Project, The Royal Institute of Technology
16. Liu C, Bouazizi I, Gabbouj M (2011) Rate adaptation for adaptive http streaming. In: Proceedings of the second annual ACM conference on multimedia systems. ACM, pp 169–174
17. Zhou B, Wang J, Zou Z, Wen J (2012) Bandwidth estimation and rate adaptation in http streaming. In: 2012 international conference on computing, networking and communications (ICNC). IEEE, pp 734–73
18. Thang TC, Ho QD, Kang JW, Pham AT (2012) Adaptive streaming of audiovisual content using mpeg dash. IEEE Transactions on Consumer Electronics 58(1)
19. Miller K, Quacchio E, Gennari G, Wolisz A (2012) Adaptation algorithm for adaptive streaming over http. In: 2012 19th international packet video workshop (PV). IEEE, pp 173–178
20. Huang TY, Johari R, McKeown N (2013) Downton abbey without the hiccups: buffer-based rate adaptation for http video streaming. In: Proceedings of the 2013 ACM SIGCOMM workshop on future human-centric multimedia networking. ACM, pp 9–14
21. Huang TY, Johari R, McKeown N, Trunnell M, Watson M (2014) Using the buffer to avoid rebuffers: evidence from a large video streaming service. arXiv:1401.2209
22. Huang TY, Johari R, McKeown N, Trunnell M, Watson M (2015) A buffer-based approach to rate adaptation: evidence from a large video streaming service. ACM SIGCOMM Computer Communication Review 44(4):187–198
23. Zhou Y, Duan Y, Sun J, Guo Z (2014) Towards simple and smooth rate adaption for vbr video in dash. In: 2014 IEEE visual communications and image processing conference. IEEE, pp 9–12
24. Nguyen HN, Vu T, Le HT, Ngoc NP, Thang TC (2015) Smooth quality adaptation method for vbr video streaming over http. In: 2015 international conference on communications, management and telecommunications (ComManTel). IEEE, pp 184–188
25. Petrangeli S, Famaey J, Claeys M, Latré S, De Turck F (2016) Qoe-driven rate adaptation heuristic for fair adaptive video streaming. ACM Transactions on Multimedia Computing, Communications, and Applications (TOMM) 12(2):1–24
26. Juluri P, Tamarapalli V, Medhi D (2015) Sara: segment aware rate adaptation algorithm for dynamic adaptive streaming over http, pp 1765–1770
27. Vo N-S, Duong TQ, Guizani M (2016) Qoe-oriented resource efficiency for 5g two-tier cellular networks: a femtocaching framework. In: Proc. of IEEE global communications conference (GLOBECOM'16), Washington, DC, pp 1–6
28. Vo N-S, Duong TQ, Hoang T, Kortun A (2017) Optimal video streaming in dense 5g networks with d2d communications. IEEE Access 6:209–223
29. Vo N-S, Nyugen TH, Nguyen HK (2018) Joint active duty scheduling and encoding rate allocation optimized performance of wireless multimedia sensor networks in smart cities. ACM/Springer Mobile Networks & Applications 23(6):1586–1596
30. Vo N-S, Duong TQ, Shu L (2012) Bit allocation for multi-source multi-path video streaming in vod systems over wireless mesh networks. In: Proc. of IEEE international communications conference (ICC12), Ottawa, Canada
31. Duong TQ, Vo N-S, Nyugen TH, Guizani M, Shu L (2015) Energy-aware rate and description allocation optimized video streaming for mobile d2d communications. In: Proc. of IEEE international communications conference (ICC15), London, UK
32. Vo S.-N., Duong TQ, Zepernick H-J et al (2011) Cross-layer design for video replication strategy over multihop wireless networks. In: Proc. of IEEE international communications conference (ICC11), Kyoto, Japan
33. Jin X, Ju H, Cho S, Mun B, Kim C, Han S (2016) Qos routing design for adaptive streaming in software defined network. In: 2016 International symposium on intelligent signal processing and communication systems (ISPACS). IEEE, pp 1–6

34. Cetinkaya C, Karayer E (2014) Sdn for segment based flow routing of dash. In: 2014 IEEE fourth international conference on consumer electronics (ICCE-Berlin), pp 74–77
35. Nam H, Kim KH, Kim JY, Schulzrinne H (2014) Towards qoe-aware video streaming using sdn. In: 2014 IEEE global communications conference. IEEE, pp 1317–1322
36. Bentaleb A, Begen AC, Zimmermann R (2016) Sdndash: improving qoe of http adaptive streaming using software defined networking. In: Proceedings of the 24th ACM international conference on multimedia. ACM, pp 1296–1305
37. Nguyen MT (2019) An energy-efficient framework for multimedia data routing in internet of things (iots). EAI Endorsed Trans Industrial Networks and Intelligent Systems 6(19):1–8
38. Team M (2019) Mininet. http://mininet.org/. Accessed 9 June 2019
39. Floodlight: opensource software for software defined networking. http://www.projectfloodlight.org/floodlight/. Accessed 9 June 2019
40. Cisco: comparing traffic policing and traffic shaping for bandwidth limiting. https://www.cisco.com/c/0Aen/us/support/docs/quality-of-service-qos/qos-policing/0A19645-policevsshape.html. Accessed 9 June 2019
41. Bitmovin: libdash - bitmovin. https://github.com/bitmovin/libdash. Accessed 9 June 2019
42. Studio OOMP (2019) Elephants dream. https://orange.blender.org/. Accessed 9 June 2019
43. Truong T-H, Nguyen H-T, Nguyen TH, et al. (2012) On relationship between quality of experience and quality of service metrics for ims-based iptv networks. In: Proc. of IEEE RIVF international conference on computing & communication technologies, research, innovation, and vision for the future, Ho Chi Minh City, Vietnam
44. Truong T-H, Nguyen HT, Nguyen TH, et al. (2013) Qoe-aware resource provisioning and adaptation in ims-based iptv using openflow. In: Proc. of 19th IEEE workshop on local & metropolitan area networks (LANMAN), Brussels, Belgium
45. Yitong L, Yun S, Yinian M, Jing L, Qi L, Dacheng Y (2013) A study on quality of experience for adaptive streaming service. In: 2013 IEEE international conference on communications workshops (ICC). IEEE, pp 682–686
46. Claeys M, Latré S, Famaey J, Wu T, Van Leekwijck W, De Turck F (2013) Design of a q-learning-based client quality selection algorithm for http adaptive video streaming. In: Proceedings of the 2013 workshop on adaptive and learning agents (ALA), Saint Paul (Minn.), USA, pp 30–37