Proceedings of the Federated Conference on
Computer Science and Information Systems pp. 819–826

DOI: 10.15439/2015F333
ACSIS, Vol. 5

An SDN-assisted System Design for Improving Performance of SVC-DASH

Cihat Cetinkaya, Yalcin Ozveren, Muge Sayit

Ege University
International Computer Institute
Izmir, Turkey
Email: {cihat.cetinkaya, muge.fesci} @ege.edu.tr;
yalcinozveren@gmail.com

***Abstract*— Today, the most of the video streaming system provides quality adaptation and prefers to send their packets over HTTP. MPEG group has standardized Dynamic Adaptive HTTP Streaming (DASH) regarding this tendency on the adaptive HTTP streaming. Besides providing quality adaptation with a non-scalable codec, DASH standard also allows Scalable Video Coding (SVC) to adapt quality. Software Defined Networks (SDN) is a recently emerged networking paradigm. SDN enables to separate control and data plane of computer networks and hence provides flexibility to network operators to implement their own routing approaches. In this paper, we propose a system for increasing Quality of Experience (QoE) of SVC-DASH clients by utilizing SDN. Our experimental results show that the proposed system provides an increase in received video quality and decrease in outage duration and startup delay when compared to the performance of the client running over todays Internet implementing shortest path routing.**

INTRODUCTION

CISCO estimates that IP video traffic will be up to 80% of global Internet traffic in 2018 [1]. The huge demand on IP video lead researches to develop video streaming applications providing high Quality of Experience (QoE) to the clients [2]. Video streaming applications need to adapt their bitrate according to the underlying network conditions to provide good performance in terms of QoE. Popular video streaming applications such as Youtube [3], Microsoft Silverlight [4], Apple HLS [5] send video packets over HTTP and perform quality adaptation by sending HTTP requests for receiving the video partitions with different qualities. HTTP protocol is preferred since it reduces the server load by HTTP caches while providing reliable data transfer.

In HTTP adaptive streaming, video files encoded at various rates are stored in the server. Each video file, i.e. representations, is separated into the small partitions called segments. The clients observe several parameters giving indication about underlying network capacity, such as buffer level or download duration to transfer the recent segments. As a result of observed parameters, the clients send their request indicating the selected representation for each segment. Hence, the selection of video quality changes over time regarding the client requests. Recently, MPEG group has standardized Dynamic Adaptive HTTP Streaming (DASH) [6]. DASH standard defines the formats of the segments and document containing the segment information.

With a non-scalable codec, video file should be separately encoded in order to obtain different representations having different quality. If scalable video coding (SVC) is used, video file is encoded once and video sequences with different quality can be extracted from this encoded file [7]. Hence, SVC provides the storage advantage. If SVC is used with HTTP streaming, storage advantage provides cache efficiency and optimizes the bandwidth usage since only one encoded video sequence is send to the HTTP cache for a video file [8]. Using SVC with DASH allows producing video sequences with large range of representations while providing cache efficiency [8].

Software defined networking (SDN) technology decouples the control and data plane of computer networks [9]. This separation allows network operators to implement different routing algorithms designed by considering specific application needs. In an SDN domain, a centralized device, called controller, has the network topology view. The controller determines the routing paths and sends related information to the forwarding elements. SDN paradigm offers a flexible platform to implement various routing algorithms designed for different network applications.

In this study, we propose a system for improving the performance of SVC-DASH clients running over SDN. We use quality of experience parameters such as received video quality, buffer underruns -i.e. outage durations-, and startup delay to measure the performance. SDN architecture enables to select different streaming path to transfer different video layer flows. We define a method for controller to learn which layers are streamed to which clients without communicating with the clients and the server, and provide path assignments using this information. We also give the design guideline of a SVC-DASH client running over SDN.

This work is funded by the Scientific and Technological Research Council of Turkey (TUBITAK) Electric, Electronic and Informatics Research Group (EEEAG) under grant 114E409.

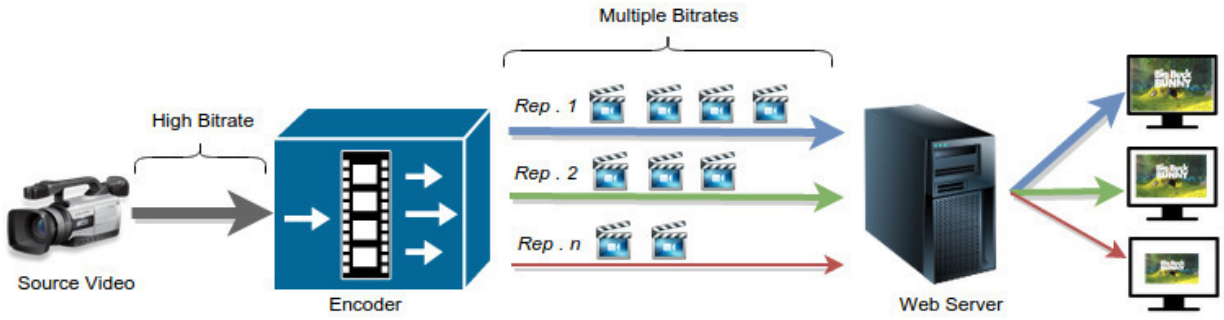


Fig. 1 DASH diagram showing different representations of the same video content.

The rest of the paper is organized as follows. In section two, the background of SDN and SVC-DASH are given with the related work proposed in the literature. In section three, the design of the proposed system is detailed. The performance of the proposed system is reported in section four. Finally, the conclusion is given followed by the references.

BACKGROUND

A. Dynamic Adaptive Video Streaming over HTTP

In DASH standard, more than one video file encoded at different bitrates for the same content is stored in a web server. In Fig. 1, a DASH diagram is given where a video content is encoded at three different bitrates with different resolutions.

In the server, each video sequence is separated into the segments. The segments can be independently decoded and has the data of fixed time interval. Information about the segments such as representation ID, URL address and bitrate are kept in Media Presentation Description (MPD) file. DASH standard defines MPD document and segment formats [6]. When a client connects to the DASH server, it gets the MPD file after establishing TCP connection between itself and the server. The quality adaptation is performed by client requests via selecting the segments with different qualities. Hence the complexity is given to the clients. DASH client decides which representation to be selected in each request. The clients adapt the video rate by changing the representations over time and send request to the server to get current segment of selected representation.

Rate adaptation algorithm is the algorithm that determines the policy of representation selection. The rate adaptation algorithms of DASH clients differ according to the parameter which is taken into account when selecting the next segment. This parameter can be measured bandwidth [10, 11] or the level of buffer fullness [12]. Some client algorithms use both of them to decide representation [13].

SVC provides to produce video sequences with various qualities in one file. The video content is encoded so as to consist of one base layer and one or more enhancement layers with SVC. While base layer can be decoded independently, it is required to base and n-1 enhancement layers to decode n^{th} enhancement layer. Base layer has the minimum quality and each enhancement layers increase the quality. In SVC-DASH or Advanced DASH, each layer corresponds to one representation. The layer dependencies are defined in MPD. Hence, when a client requests for a segment, it may also request segments of lower layers according to the dependencies given in MPD document.

B. Software Defined Networking

SDN paradigm enables to separate control and data planes of computer networks. This approach moves the intelligence from forwarding elements to an external device, called controller. Network operators can implement novel routing strategies taking requirements of network applications into account by programming the controller. The controller sends commands containing forwarding rules to the forwarding elements, i.e. switching devices. Another advantage that SDN provides is that the controller can have the network topology information such as traffic amount on the links, packet loss ratio or drop packets by communicating with the switches. The SDN architecture is given in Fig. 2.

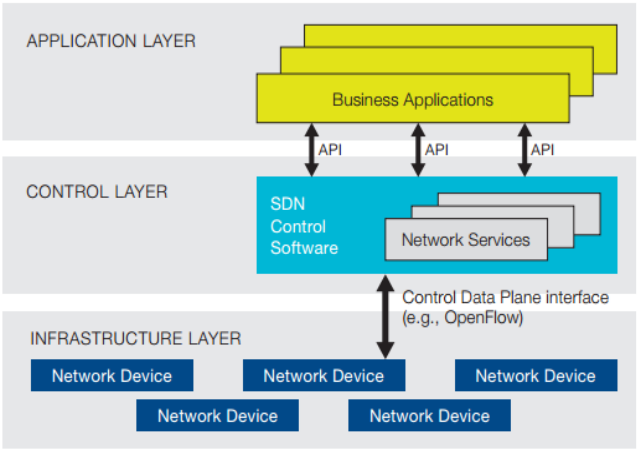


Fig. 2 SDN architecture showing decoupled structure of control plane and the data plane [9].

OpenFlow is the first protocol designed for providing communication between the controller and the switches. OpenFlow opens a secure communication channel. Switches update their forwarding tables according to the messages of the controller, sending via using OpenFlow protocol. When a switch gets a packet, it checks the header information of the packet, and performs an action regarding the defined rule for that information. In other words, the switches implements a <match, action> processing for each packet. Forwarding a packet, discarding a packet, or querying controller for asking the rules of a packet can be given as action examples.

C. Related Work

Since both DASH standard and SDN paradigm have gained great attention from the academia and market, researchers has proposed several studies related on these topics recently.

In [14], dataset for SVC-DASH is presented. HTTP cache efficiency of SVC-DASH is discussed in [8]. In this study, it is shown that if SVC-DASH is used, then network traffic reduces when compared to the traffic amount of MPEG-DASH.

The performance in terms of QoE of DASH clients can be improved by utilizing SDN capabilities. In [15], the authors aim the provide fairness among DASH clients. For this purpose, the controller communicates with the clients and sends commands indicating which representation should be selected. This selection is determined according to the network topology information and client type. Since SDN provides controller to decide forwarding rules, different selection strategies for the flow routes between server and the clients is also proposed in several studies to increase QoE of DASH clients. In [16], a traffic shaping policy is determined for OpenFlow enabled switches, which gives priority to the HTTP flows. This approach provides DASH clients to achieve higher performance when compared to the case that HTTP flows has no priority. QoE centric path assignment over SDN networks for DASH clients is proposed in [17]. The controller communicates with DASH clients and if the performance decreases in terms of QoE parameters, the bottleneck point is detected by the information obtained from the switches. If the problem is in SDN network, then a different path is selected, otherwise the server is changed [17]. Path assignment by considering the bitrate of the segments and path capacities for increasing DASH clients' performance running over SDN networks is proposed in [18]. The server signals the controller to give information about the bitrate of the current segment and the paths are changed if the capacity of the current path is not enough to convey that segment packets.

Path assignment capability of SDN networks brings a natural approach to send packets of scalable coded video by sending base layer video over a different path. In [19], base layer video packets is send over a lossless path while the remaining traffic is send over the shortest path over SDN network. Besides sending base layer video packets over lossless path, forwarding enhancement layer packets over another path is proposed in [20]. The flow routes of base and enhancement layers packets are determined according to the output of an optimization model by considering packet loss and delay variation [21]. Quality adaptation techniques and flow route decisions considering the quality adaptation are not briefly discussed in these studies.

To the best of our knowledge, there is one approach proposed in the literature to route HTTP flows for SVC-DASH clients running over SDN. In [22], the path that has the smallest load is assigned to route base layer packets. Similar to base layer path assignments, the paths determined for enhancement layers are done based on layer priority; the packets of enhancement layer having higher priority are sent over a path having smaller load. The SVC-DASH client always requests all enhancement layers for each segment, i.e. it has no rate adaptation algorithm. Since rate adaptation is not implemented in that client software, there is no strategy defined for selecting the representation to be requested. The client sends requests in every 2 seconds [22].

In this study, we propose a system for SVC-DASH streaming over SDN. In the system, we think each video layer packets as a separate flow and assign streaming paths considering the bitrate of the layers. The paths of each layer can be the same if the available bandwidth is enough, but can be different, otherwise. Our approach differs from the studies in the literature in the following aspects. First, the controller does not communicate with the server or the clients to get information about the client requests. Second, we utilize different OpenFlow message types to detect received quality. And finally, we give an implementation guideline for SVC-DASH clients running over SDN.

FLOW ROUTE ASSIGNMENT FOR SVC-DASH OVER SDN

In this section we give the details of the proposed system for improving the performance of SVC-DASH clients running over SDN and give the details of SVC-DASH client implementation in the proposed system.

D. SDN for SVC-DASH Clients

In the proposed system, flow routes i.e. streaming paths between the server and the clients are determined by the controller. The controller communicates with the switches in both proactive and reactive manner. It queries the switches periodically in order to obtain traffic information as proactive behavior. The controller calculates the available bandwidth of each path in the network by using the traffic information. Whenever the controller receives a message from a switch querying for a flow, it sends the forwarding rule to that switch. This is the reactive part.

Since the controller determines the paths for video flows between server and the clients, the switches need to communicate with the controller to learn the forwarding

rules. The controller determines the paths for video layer packets; and then sends the forwarding rules related to the assigned path. Note that, the controller has to send the message containing forwarding rules to all switches along the chosen path.

As a path assignment strategy, the controller can assign the path having maximum available bandwidth for the newly joined client. Although the path having maximum available bandwidth is selected as streaming path for that client, sending each video layer packets over the same streaming path may decrease the capacity utilization. For example, suppose we have two paths where the capacity of first and second path equal to bitrate of base and first enhancement layer, respectively. In this case base layer packets can be sent over the first path and enhancement layer packets can be sent over the second path. But capacity of each path is not enough to transfer both layers packets.

Because of capacity utilization problem, we advocate to assign different streaming paths for the video layer packets requested by the same client for maximizing the network capacity utilization. For capacity utilization maximization, the best solution is to assign paths regarding the bitrate of the layers being sent. When a client joins the system, the controller makes system-wide path assignment for each client in the system by taking capacity of the paths and bitrates of the video layers into account.

In order to assign paths for the video layers packets of the clients, one approach is for controller to communicate with the clients or the server, to obtain requested layer information and to assign path according to the retrieved layer information. However, this reactive approach brings extra message complexity to the system. Instead of communicating with the server or the clients, we propose a method for controller to have the requested layer information without introducing extra messaging burden. We provide that the server sends each layer from a different TCP port and put this port information of video layers into the MPD file. The clients send video layer requests according to the given information in the MPD file. For example, if a client requests base and one enhancement layer, it sends these requests to defined ports of base and first enhancement layers, not the same port on the server. Hence, the switches look for a match of the tuples (client IP address, server IP address, port number) when receiving request messages of the clients. If this match does not present in its flow table, the switches sends a query containing the tuple.

SVC-DASH client establishes a TCP connection between the server and itself after connecting to the system. The first message of the TCP connection establishment is TCP SYN message being sent by the client. The OpenFlow enabled switch that newly joined client is connected to sends a PACKET_IN message to the controller after receiving TCP SYN message. The controller assigns paths upon getting a PACKET_IN message. Since the controller has not the information of what representation, i.e. which layers will be requested by the client; it determines the paths for base layer packets and for enhancement layer packets in case of there is enough network capacity.

```
Procedure Path_Assignment

1:   c_i: i^th online client
2:   N: number of online clients
3:   cp_p: the capacity of path p

4:   i = 0;
5:   while i<N
6:      p: the path having max available bandwidth
7:      assign p for the base layer packets of client i
8:      cp_p = cp_p – bitrate of base layer;
9:      i++;
10:  end while

11:  i = 0;
12:  current enhancement layer = first enhancement layer;

13:  while current enhancement layer ≠ NULL && there
 exist any p with cp_p ≥ bitrate of current enhancement layer

14:     while i<N
15:        p: the path having max available bandwidth
                                              (i.e. capacity)
16:        if (cp_p ≥ bitrate of current enhancement layer)
17:           assign p for the current enhancement layer
                          packets of client i
18:           cp_p = cp_p – bitrate of current enhancement layer;
19:        end if
20:        i++;
21:     end while

22:     if (current enhancement layer =
                                highest enhancement layer)
23:        current enhancement layer = NULL
24:     else
25:        current enhancement layer = next enhancement
                                                   layer;
26:     end if
27:  end while
```

Fig. 3 Path assignment algorithm running by the controller. The output of the algorithm determines the paths for all clients

The path assignment algorithm is given in Fig. 3. As given in the algorithm, the controller first assigns the paths having maximum available bandwidth for transferring base layer packets of all clients between the 5th and 10th lines of the algorithm. Since base layer packets are crucial, the controller has to assign paths for them even if the capacity of the network is not enough to send these packets properly. Upon completing the path assignment for the base layer packets, the controller assigns the paths for enhancement layer flows starting from the first enhancement layer for all clients in the while loop starting at line 14. But this time, if the network

capacity is not enough to transfer the packets of an enhancement layer, then this assignment procedure is stopped. In other words, the paths for enhancement layer packets are assigned if and only if there is enough capacity.

After determining the paths for transferring the packets of video layers, the controller sends related flow information to the switches. The switches overwrite new forwarding rules to the existing entries in their flow tables. Note that at this point, although the switches have the information of rules for forwarding base and one or more enhancement layer packets, the clients may not request to receive any enhancement layer, in turn, these rules may not be used. Here, the controller determines the paths in proactive manner.

As defined in HTTP adaptive streaming, the clients request for a representation after it joins the system and gets MPD file from the server. Besides that, online clients continue to request representations according to their rate adaptation algorithms. At this point, there are three possible cases for a client:

(*i*) The controller may have assigned the paths for exactly same number of enhancement layers with the number of enhancement layers that a client requests to.

(*ii*) The controller may have assigned paths for *i* enhancement layers but the client requests more than *i* enhancement layers packets.

(*iii*) The controller may have assigned paths for *i* enhancement layers but the client requests less than *i* enhancement layers packets.

Clearly, for the first case, the forwarding rules for the requested representations, i.e. video layers flows are already determined on the flow tables of the switches.

In the second case, the switch gets a packet (or more than one packet) which flow information does not present in its flow table. It sends a PACKET_IN message to learn what to do with the received packet. This case occurred when the client measures available bandwidth and decides that it can receive more enhancement layers, which shows that the available bandwidth of at least one of the paths assigned for that client is higher than expected. Suppose the client receives video packets over n paths. The controller assigns the path having maximum available bandwidth among these n paths for the requested flow.

In the third case, the switches have more forwarding rule entry than it is required to. In this case, there are unused flow entries in the flow tables of the switches. The same situation can happen when a client decreases the quality by requesting less number of enhancement layers than that of it requested in previous period. In the OpenFlow protocol, each flow entry has an idle timeout which is determined by the controller. The switch removes a flow which has no matching packets in idle timeout and informs controller by sending FLOW_REMOVED message. In this work, the flow of base layers has no idle timeout where the enhancement layers is set to a certain threshold called flow timeout. When controller receives a FLOW_REMOVED message, it determines the path having minimum throughput among the paths which are assigned to transfer video layer packets for that client. The controller changes this path to a new path having maximum available bandwidth and sends related forwarding rules to the corresponding switches. The controller performs the second step in order to find better path which can be provide to increase the quality received by the client.

There is one thing left subtle but important. As stated in previous section, DASH clients request for the segments consecutively. If a segment is downloaded fast, the client waits for a while before it sends the request for the next segment. In other words, it enters the successive downloading and waiting periods. This phenomenon is known as on/off behavior [23]. On/ off period cause a traffic measurement problem if the controller measures of a current flow traffic when the client is in off period. If the client is in off period, then the controller measures the current traffic on that path as zero. In order to prevent that misconclusion, we have conducted a set of experiments and determine maximum length of off periods. Then we set idle timeout of the flows to a value higher than maximum off period length. Hence, we provide that the controller to sense traffic if there is at least one client receiving packets over corresponding path.

E. An Implementation of SVC-DASH Client Running over SDN

Typically, two parameters are determined regarding the output of the rate adaptation algorithm: which representation (if SVC is used, which enhancement layers) will be requested and when it will be requested. In this section, we give the details of the implemented SVC-DASH client software.

When a client joins the system, it requests MPD file from the server as stated in DASH standard. Since we added the port information in MPD file, the client extracts the port information for each video layer from the MPD document besides the information of the bitrate of each video layer. The client also estimates the available bandwidth by calculating the download rate of the MPD document. Based on this calculation, the client determines the number of video layers to be requested for the first segment.

In SVC-DASH client implementation given in [22], the client requests base and enhancement layers segments consecutively. For example, if client decides to request video segment of second enhancement layer after executing rate adaptation algorithm; it sends request for base layer packets first. Upon receiving the base layer packets, it requests for the first enhancement layer packets, and after receiving first enhancement layer packets, it requests second enhancement layer packets.

Since the controller can assign different paths to the video layer flows in SDN network, the clients does not have to request video layers sequentially. In our SVC-DASH client

implementation, the client requests video layers simultaneously after deciding which layers will be requested. Suppose the client decides to request one base and two enhancement layers. The clients send an HTTP GET message to the corresponding ports of the server at the same time. Hence, the client throughput is maximized since it does not have to wait downloading packets of a video layer to request the next layer packets for a segment.

In order to calculate throughput, we need a new method since each video layer packet transmitted over a different path. Suppose the client request base and n enhancement layers. Let t_{dt} represents the total time to download base and these n enhancement layers. Then t_{dt} and the throughput for the requested segment are calculated according to the formula given in (1) and (2), respectively. In the formulas, e_i represents the i^{th} enhancement layer, b represents base layer. t_x^{last} is the receiving time of the last packet of x^{th} layer, $t_{request}$ is the sending time of the request and br_x is the bitrate of the x^{th} layer.

$$t_{dt} = \max(t_b^{last}, t_{e_1}^{last}, \ldots, t_{e_n}^{last}) - t_{request} \quad (1)$$

$$\frac{br_b + \sum_{i=1}^{n} br_{e_i}}{t_{dt}} \quad (2)$$

After calculating the throughput, the client selects the representation which has the maximum bitrate value lower than the throughput. The client sends next HTTP GET message upon downloading the requested layers packets of current segment.

PERFORMANCE EVALUATION

F. Experimental Setup

The experiments have been performed over a test bed shown in Fig. 4. SDN environment including OpenFlow-enabled switches and hosts that run SVC-DASH clients and SVC-DASH server are emulated using Mininet [24] software. SDN controller is implemented using Floodlight [25] software.

In the experiments, Big Buck Bunny video [14] which has one base layer and three enhancement layers are served by the SVC-DASH server. The bitrates of the video layers are given in Table 1. Note that the bitrates given in the table are cumulative because of inter-layer dependency. The video is divided into 300 segments each with a length of 2 seconds. Therefore the experiments last for 600 seconds. The clients join the system with interval of 30 seconds and never exit during the experiments. Flow idle timeout is set to 20 seconds.

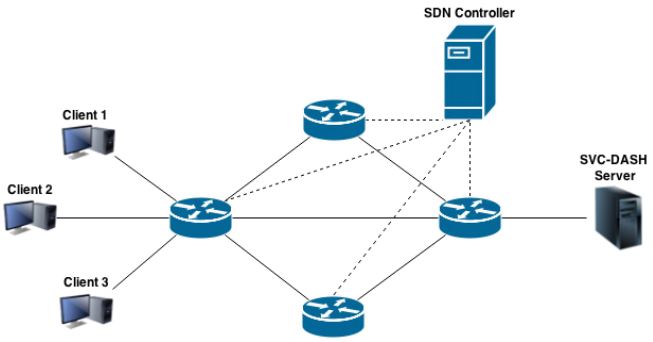


Fig 4. Experimental Testbed

TABLE I.
THE BITRATE DISTRIBUTION OF VIDEO LAYERS

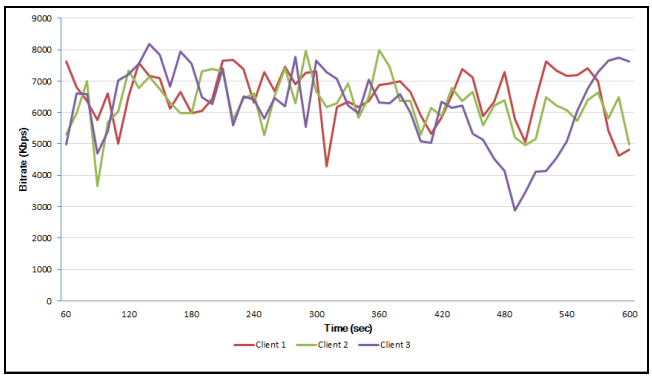
Video Layer	*Bitrate*
Base	1555 Kbps
Base + Enhancement 1	2700 Kbps
Base + Enhancement 1 + Enhancement 2	4547 Kbps
Base + Enhancement 1 + Enhancement 2 + Enhancement 3	6857 Kbps

As seen from Fig. 4, there are three different paths between server and the clients in the network topology. The capacity of each path in the network equals to 6000 Kbps. This means that although the network capacity is enough to send base layer to each client, there is not enough capacity to send the highest enhancement layer to all clients in the system.

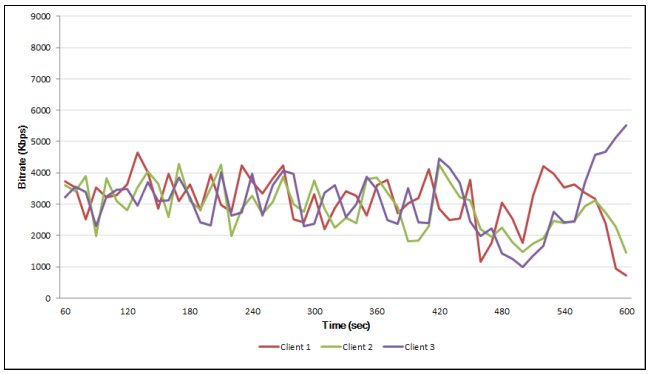
G. Experiment Results

In order to show performance of the proposed system, we observe several QoE parameters: received bitrate, received video quality and, number and length of durations. We also perform experiments with same set of parameters by using traditional Internet shortest path routing and give the results comparatively. The topology includes only one shortest path with the length of one hop. Hence, the video packets are streamed over the same path with shortest path routing. Each experiment is repeated 10 times; averaged results are obtained and shown in the graphs.

In Fig. 5a and Fig. 5b, the average bitrates received by each client in the proposed system and in the system with shortest path routing are given as a function of time, respectively. While the averaged received bitrate values for the clients are in the range of 5000 Kbps and 8000 Kbps, these values are in the range between 2000 Kbps and 4000 Kbps with shortest path routing. In both approaches, the clients can achieve bitrate higher than the capacity of the paths. The main reason of that is on/off period pattern of the clients. A client may even use the full capacity of a path if other clients using that path are in their off periods.



(a). Received bitrates with the proposed system



(b). Received bitrates with the shortest path routing

Fig 5. Averaged received bitrate as a function of time

In order to show the quality of the received video, we measured the total length of the video segments that received from each layer and give the results in Fig. 6. In the figure, base and enhancement 3 denotes the minimum and maximum quality, respectively. During the streaming session, the clients in the proposed system experience the video quality provided by the enhancement 2 most of time, where the clients receive the base layer quality in the shortest-path approach. Furthermore, the clients in the shortest-path approach never experience the quality provided by the enhancement 3. Note that, the values given in the graph are cumulative. In other words, if receiving an enhancement layer is shown in the graph, it means base and all enhancement layers below the received enhancement layer are also received.

When the bandwidth is not enough to send the selected representation, the clients may drain their buffers and then start to experience outages. In Fig. 7, average durations in seconds observed in the proposed system and the shortest path approach are given. As it is seen, the proposed system has a lower number of outages. The reason of this is that the proposed system dynamically changes the paths and provides to increase network capacity utilization. Thus the clients get the maximum available bandwidth. But in the shortest path approach, the clients always use the same path. So their share of bandwidth decreases which causes outages to increase. In Fig. 8, CDF graph of total outage durations is given in order to show the distribution of outage durations.

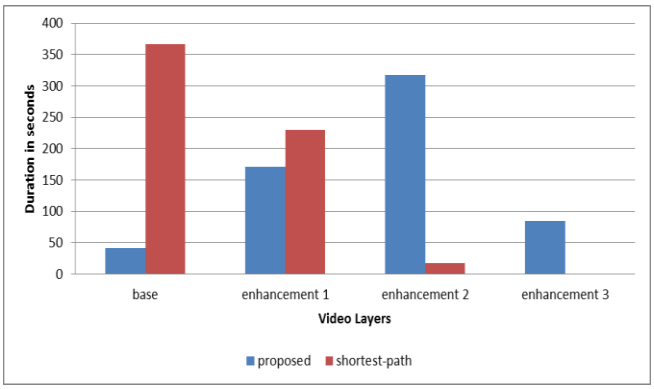


Fig 6. Received video quality in terms of layers

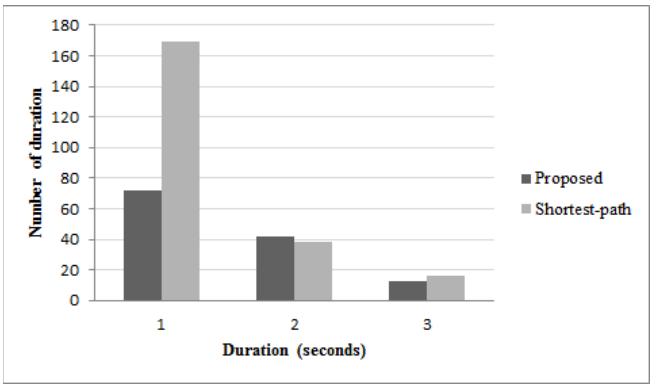


Fig 7. Total duration values of outages measured in the experiments

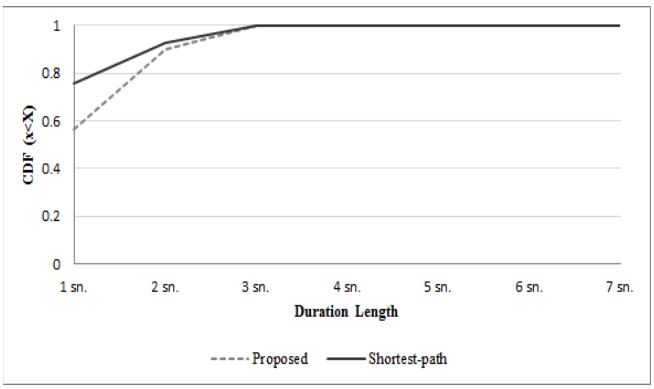


Fig 8. CDF graph of observed duration of outages

In Table 2, averaged startup delays experienced by each client are given. Startup delay equals to the elapsed time from requesting the first segment to playing the video. As seen from the table, limited capacity causes an increase in startup delays. The clients experience longer startup delay with shortest path routing.

TABLE II. STARTUP DELAY VALUES IN SECONDS

	Proposed system	*Shortest-path routing*
Client 1	10 sec	10 sec
Client 2	11 sec	12 sec
Client 3	11 sec	17 sec

CONCLUSION

In this paper, we propose a system for increasing QoE SVC-DASH client running over SDN. For this purpose, we think each layer of the video as a separate flow and assign paths by taking path capacity and the bitrate of the layers into account.

Our system differs from the similar studies in the literature as the controller does not communicate with the server and the clients to detect which video layers are sent to the clients. In order to detect requested number of enhancement layers, we utilize PACKET_IN and FLOW_REMOVED messages sent by the switches to the controller. The controller decides system wide re-routing of flows or change the path of a flow according to the received message type and capacity of the paths.

The performance of the proposed system is measured by considering the QoE parameters such as received video quality, outage duration and startup delay. When we compare the achieved performance of the proposed system with the traditional shortest-path routing of today's Internet, our system provides to yield better performance for each QoE parameter.

In the future, we are planning to measure the performance of the proposed system with larger number of clients, different types of video resolutions and with different types of network topologies.

REFERENCES

[1] Cisco VNI, "Cisco Visual Networking Index: Forecast and Methodology, 2013–2018", White Paper, 2014.

[2] M. Dąbrowski, "Emerging technologies for interactive TV", in proc. of the Federated Conference on Computer Science and Information Systems (FedCSIS), pp. 787–793, 2013.

[3] http://www.youtube.com

[4] Microsoft, IIS Smooth Streaming Transport Protocol, Sept. 2009; https://msdn.microsoft.com/en-us/library/ff469518.aspx/[MS-STTR].pdf.

[5] https://developer.apple.com/streaming/

[6] I. Sodagar, "The MPEG-DASH standard for multimedia streaming over the Internet", IEEE Multimedia, pp. 62–67, 2011.

[7] H. Schwarz, D. Marpe, and T. Wiegand, "Overview of the scalable video coding extension of the H.264/AVC standard," IEEE Transactions on Circuits and Systems for Video Technology, vol. 17, no. 9, pp. 1103–1120, 2007.

[8] Y. Sanchez, C. Hellge, W.V. Leekwijck, Y.L. Louédec, and T. Schierl, "Scalable Video Coding based DASH for efficient usage of network resources", Position Paper for the Third W3C Web and TV workshop, Los Angeles, CA, USA, September 2011.

[9] Open Networking Foundation (ONF), "Software defined networking: the new norm for networks", White paper, 2012.

[10] J. Jiang, V. Sekar, and H. Zhang, "Improving fairness, efficiency, and stability in HTTP-based adaptive video streaming with FESTIVE", in proc. of the 8th international conference on Emerging networking experiments and technologies (CoNEXT), 2012.

[11] B. Rainer, S. Lederer, C. Müller, and C. Timmerer, "A seamless Web integration of adaptive HTTP streaming", in proc. of EUSIPCO, 2012.

[12] T.Y. Huang, R. Johari, and N. McKeown., "Downton abbey without the hiccups: Buffer-based rate adaptation for HTTP video streaming", in proc. of the ACM SIGCOMM workshop on Future human-centric multimedia networking (FhMN), NY, USA, 2013.

[13] L. D. Cicco, V. Caldaralo, V. Palmisano, and S. Mascolo, "ELASTIC: A client-side controller for dynamic adaptive streaming over HTTP (DASH)", in proc. of the IEEE 20th international Packet Video Workshop (PV), 2012.

[14] C. Kreuzberger, D. Posch and H. Hellwagner, "A scalable video coding dataset and toolchain for dynamic adaptive streaming over HTTP", in proc. of the ACM MMSys, Portland, Oregon, 2015.

[15] P. Georgopoulos, Y. Elkhatib, M. Broadbent, M. Mu, and N. Race, "Towards network-wide QoE fairness using OpenFlow-assisted adaptive video streaming", in proc. of the ACM SIGCOMM workshop on Future human-centric multimedia networking, 2013.

[16] M. S. Seddiki, M. Shahbaz, S. Donovan, S. Grover, M. Park, N. Feamster, and Y. Song, "FlowQoS: QoS for the rest of us", in proc. of the hotSDN workshop, 2014.

[17] H. Nam, K. Kimy, J. Y. Kimy, and H. Schulzrinne, "Towards QoE-aware Video Streaming using SDN", in proc. of the GLOBECOM, 2014.

[18] C. Cetinkaya, E. Karayer, M. Sayit, and C. Hellge, "SDN for Segment based Flow Routing of DASH", in proc. of the IEEE 4th International ICCE conference, 2014.

[19] S. Civanlar, M. Parlakisik, A.M. Tekalp, B. Gorkemli, B. Kaytaz, and E. Onem, "A QoS-enabled OpenFlow environment for Scalable Video streaming", in proc. of the IEEE GLOBECOM Workshops, pp. 351-356, 2010.

[20] H.E. Egilmez, B. Gorkemli, A.M. Tekalp, and S. Civanlar, "Scalable video streaming over OpenFlow networks: An optimization framework for QoS routing", in proc. of the 18th IEEE International Conference on Image Processing (ICIP), pp. 2241-2244, September 2011.

[21] 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.

[22] S. Laga, T. Van Cleemput, F. Van Raemdonck, F. Vanhoutte, N. Bouten, M. Claeys, and F. Amd De Turck, "Optimizing scalable video delivery through OpenFlow layer-based routing", in proc. of the IEEE Network Operations and Management Symposium (NOMS), 2014.

[23] Z. Li, X. Zhu, J. Gahm, R. Pan, H. Hu, A. Begen, and D. Oran, "Probe and adapt: Rate adaptation for HTTP video streaming at scale", IEEE J. on Selected Areas in Comm., vol. 32, no. 4, pp. 719-733, 2014.

[24] B. Lantz, B. Heller, and N. McKeown, "A network in a laptop: Rapid prototyping for software-defined networks", in proc. of the ACM SIGCOMM Workshop on Hot Topics in Networks, pp. 1-6, USA, 2010.

[25] Floodlight. http://www.projectfloodlight.org/floodlight/.