ELBA: Efficient Layer Based routing Algorithm in SDN

Ankit Gangwal*, Megha Gupta*, Manoj Singh Gaur*, Vijay Laxmi*, Mauro Conti†
* Department of Computer Science & Engineering, Malaviya National Institute of Technology, India.
† Department of Mathematics, University of Padua, Italy.
Email: {2014pcp5290, 2014pcp5026, gaurms, vlaxmi}@mnit.ac.in, conti@math.unipd.it

***Abstract*—Adaptive streaming dynamically adapts video quality level according to the perceived device status and network conditions. It requires several representations of the same content, each encoded at different quality rates. As a representative example, H.264/SVC eliminates the requirement of redundant representations, improving the efficiency of caching and storage infrastructure. SVC video consists of a "Base Layer" and one or more of "Enhancement Layers". These layers have inter-dependencies and different QoS requirements. On another side, SDN allows forwarding tables to be adjusted dynamically, enabling us to route every individual flow differently. In this paper, we propose ELBA, an algorithm for scalable video streaming over SDN. ELBA utilizes the dynamic re-routing capability of SDN, to stream different layers of SVC coded video over possibly different suitable paths. In the proposed video streaming system, we use a novel mechanism to exchange information between the control plane and streaming servers. We have compared the performance of our approach with traditional Internet routing technique. Our evaluation results show that our proposal is not only feasible but in particular, it significantly outperforms the traditional Internet routing approach in terms of QoE.**

***Index Terms*—H.264/SVC, QoE, SDN, Video Streaming**

I. Introduction

Over the past decade, the demand for multimedia services over the Internet has witnessed a tremendous growth. Internet video traffic is projected to be 80% of entire Internet traffic by 2019 [1]. Web video streaming, mobile TV, real-time video conference and many other streaming media applications require steady network resources with no or little variations. These specific requirements cannot always be effectively met by the best-effort Internet. Also, there may exist more than one route (path) between network entities, where each path may have different properties, e.g., one path may provide lower error rate while the other may offer higher bandwidth. For a reliable transmission of video streams and reuse of existing infrastructure adaptive streaming techniques are gaining popularity.

In adaptive streaming, a video is split into segments, and different quality representations are used to encode these segments. Traditionally, H.264 Advanced Video Coding (H.264/AVC) [2] is used to encode video segments. However, AVC lacks scalability, restricting it to meet the diverse requirements of different users having varying displays sizes and connected through differing network links. To overcome this limitation, a scalable extension of H.264/AVC was proposed, which is known as H.264 Scalable Video Coding (H.264/SVC)

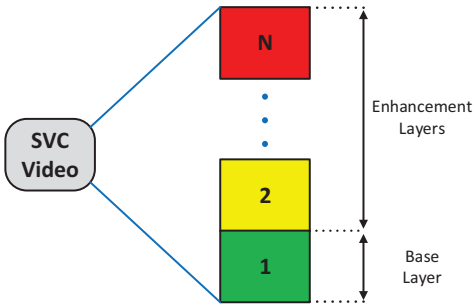

Figure 1: Illustration of SVC video layers

[3]. SVC encodes a video in a base layer and one or more enhancement layers, as shown in Figure 1. The base layer provides basic (standard) video quality while adding enhancement layer to the base layer enhances quality.

Software-defined networking (SDN) [4, 5] is a recently emerging paradigm. The key concept of SDN is to decouple data plane and control plane. The data plane provides actual forwarding functionality. The data plane is controlled by the control plane. The control plane is programmable. The programmability of control plane enables us to devise flexible routing algorithms, which can adapt according to the diverse requirements of various network applications.

The aim of developing a new routing algorithm is to enhance the quality of experience (QoE) for the end user. There are various proposals for materializing new streaming technique for video applications working over SDN. But one commonly practiced assumption in SDN literature is that the control plane always has prior information about the characteristics (e.g. bit-rate, etc.) of the video traffic. One of the biggest challenges in SDN environment is to efficiently exchange traffic's characteristics information between hosts and the control plane.

In this paper, we propose ELBA (Efficient Layer Based routing Algorithm), an algorithm that exploits the dynamic re-routing capability of SDN, to stream different layers of SVC coded video over distinctly suitable paths. It intends to improve the delivered video quality without affecting rest of the network traffic and also to improve utilization of network resources. We also propose a novel and feasible mechanism to exchange information between the control plane and streaming servers. To the best of our knowledge, a mechanism to exchange information about network traffic's characteristics

978-1-5090-2279-3/16/$31.00 ©2016 IEEE

between the control plane and data plane elements has not been proposed previously. While in this paper we focus particularly on scalable videos, we believe that the approach used in this context can be extended to other network services as well.

The remainder of this paper is organized as follows. Section II presents an overview of related work regarding multimedia communications over SDN. Section III elaborates the proposed video streaming system and routing algorithm. The details of prototype implementation and results are discussed in Section IV. In Section V, the conclusions are given, followed by the references.

II. Related Work

In SDN environment, the key concept is to partition a network into data plane and control plane. The data plane consists of many forwarding devices that provide actual forwarding of data packets. The data plane is controlled by the control plane. The control plane consists of at least one decision-making entity called the controller, which has a global view of the network of its domain. The controller communicates with the forwarding devices to acquire traffic information in real-time. Utilizing topology information and the real-time traffic statistics, the controller can decide the route of data packets in the network. Hence, SDN allows the network operators to develop application-specific route decision strategies considering the network topology information and the obtained traffic statistics.

The controller and the switches communicate via OpenFlow [6] protocol. It creates a secure communication channel between the controller and switches. The controller instructs the switches by simply updating their 'flow table' via the OpenFlow protocol. Apart from adding flow table entries, the controller can also delete/modify existing entries. It can also ask for real-time traffic statistics using the OpenFlow protocol.

In [7] and [8], authors have addressed that overall network throughput can be significantly affected by Quality-of-Service (QoS) routing. A routing algorithm striving to optimize the QoS traffic cannot ignore the presence of best-effort traffic. As the majority of the Internet traffic is best-effort, any performance enhancing approach concerned with QoS traffic must also account for best-effort traffic and prevent its congestion. To attain better performance, Peter et al. in [7] suggest that QoS traffic should avoid heavily loaded best-effort shortest-paths. The work presented in [9] proposes a model of an intermediary adaptation node, where an estimation of the available bandwidth on the client's link by media gateway removes identified SVC streams. Schierl et al. in [10] demonstrate a graceful degradation of quality in SVC streaming when the network load increases.

In [11], a flow rate shaper for SDN is proposed, which adapts the transmission pace according to the flow rate of distinct applications. Based on SDN framework, a network caching service along with load balancing for video on demand (VoD) applications is proposed in [12]. According to service negotiations, the work in [13] suggests to assign appropriate flows to the users. Patrick et al. in [14] propose to assure a predictable quality of service level. In the proposed work, an IPTV service operates in SDN system, and there are two paths between the client and server. The secondary path is chosen for video streaming whenever a problem is discovered on the primary path. In this work the method of primary path selection is not defined. In [15], the controller changes the quality of the streamed video according to the client's download capacity. In this study, client's download capacity information is acquired from the switches, but path selection method is not discussed.

A non-layered codec requires several representations of the same video content, each encoded at different quality rates. In contrast, layered codec such as H.264/SVC offers higher storage and transmission efficiency. SVC video contains a "Base Layer" and one or more of "Enhancement Layers". These layers have inter-layer decoding dependencies. The video plays at the lowest quality when the client gets only the base layer. Adding an enhancement layer enhances the quality [16].

Civanlar et al. in [17] propose to route the base layer over a lossless path. After considering the available bandwidth and congestion, the controller selects a path having adequate bandwidth to stream the base layer. In [18], the paths are allotted by analyzing the length and capacity of the path. The work in [19] proposes amendments to [18] for multi-domain SDN networks. The work in [20] considers the priorities of the base layer and enhancement layer packets to define different flow rules for these layers. Each video layer is streamed through different TCP port. The controller identifies each video layer by inspecting the TCP source port value of the packets. In this study, the process of informing the controller about this mapping of TCP port value to the video layer streamed through it is not discussed. Similar to this approach, another path selection technique for streaming the base and enhancement layers separately is discussed in [21]. In [22], a learning based approach is proposed. The controller considers the available bandwidth for route selection and performs periodic quality adaption by instructing the streaming server to add/remove one or more video layers. The controller signals the video server via its northbound API. However, the method of intimating the controller about the bit-rate information of each video layer is not addressed in this study. Similarly, the procedure of informing the controller about the characteristics of video layers is not discussed in [17, 18, 19, 21].

In this paper, we propose an efficient video streaming technique to enhance the quality of the delivered video. It uses a combination of the dynamic routing capabilities of SDN and the layered characteristics of SVC video to stream different layers of SVC coded video over possibly different suitable paths. We also propose a novel and feasible mechanism to convey information about the characteristics of video layers between the control plane and data plane elements.

III. ELBA: Proposed Scalable Video Streaming Technique

In this section, we present ELBA, our solution for scalable video streaming over SDN. Here, we elucidate the underlying principles and overview of ELBA, followed by its comprehensive implementation details.

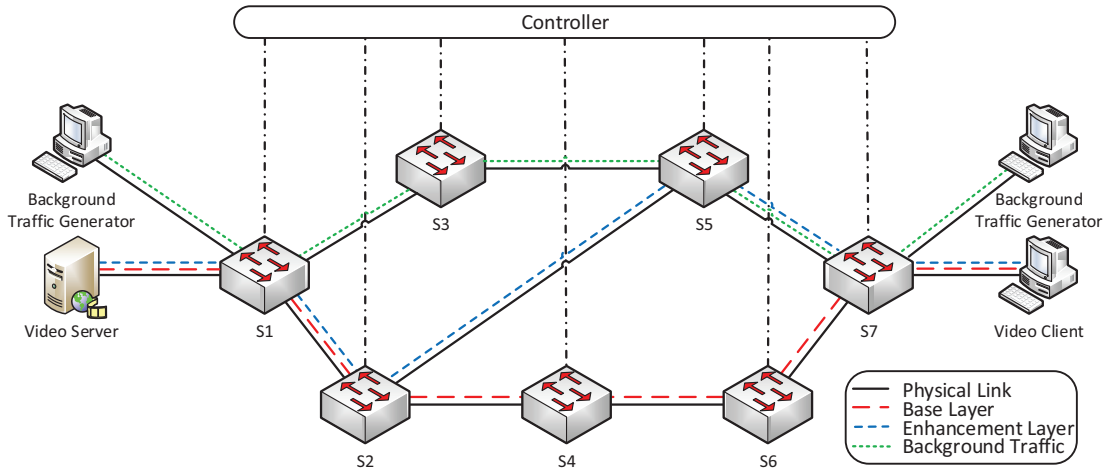

Figure 2: ELBA: an illustrative example for video streaming

A. Algorithm Design

According to the tenet of SVC, an upper layer of video becomes useless if any of the corresponding lower layers is absent at the decoding time. Hence, the base layer packets are the most important as the absence of these packet makes all enhancement layer packets useless and the video starts to freeze. In real-time streaming, retransmitting lost packets is usually not suitable. On the other side, if an enhancement layer packet is lost, the video continues to play with degraded quality. So, the traffic stream in the network can be categorized into following distinct streams:

1) As SVC base layer must be streamed without any packet loss, hence it can be defined as lossless QoS (i.e., any packet loss is intolerable) traffic.
2) All SVC enhancement layers are defined as lossy QoS (i.e., few packet losses are tolerable) traffic.
3) Background traffic can be treated as best-effort traffic.

The quality of received video at the client depends greatly upon two factors, loss and delay. Due to the loss, the video packets might reach the client either as corrupted packets or they do not reach the destination at all. While due to the delay, the video packets might not reach the destination on time. Moreover, the effect of these consequences varies with the position of a layer in the SVC layer hierarchy.

To optimize the video delivery, based on the routing decision the controller in our approach may assign distinct routes for each video layer. Paths for lossless and lossy QoS traffic is calculated using a slightly modified version of Dijkstra algorithm [23], where the respective edge's weight is considered according to Eq. 3. The dominating best-effort traffic is traditionally accommodated on a hop-based shortest-path. While in our approach, it is accommodated on a bottleneck shortest-path (maximum bandwidth) path [24], where the available bandwidth on a link serves as the capacity of the edge. Figure 2 depicts an abstract working of the proposed algorithm for a specific (and simple) topology. At S1, the controller decides to forward both base and enhancement layers over the same link until S2. At S2, both the layers take a separate path to reach the client. While the background traffic is forwarded via a different path from S1, which meets enhancement layer at S5. Influence of the link loss and link delay is elaborated in the remaining section.

1) Influence of the loss: Due to packet losses, exact frames cannot be decoded at the receiver's side. Furthermore, an undecoded lower layer makes all the corresponding upper layers useless, and the video starts to freeze. For this reason, higher priority is given to lower video layers, and lower layers are streamed over a path having a higher probability of delivering these layers. To accomplish this behavior, the loss component W_{loss} of the edge weight is weighted by a priority factor v_i for every i^{th} video layer. The value of v_i is inversely proportional to the priority of an i^{th} video layer. The loss component W_{loss} of the edge weight is given by Eq. 1, where L is the loss probability on the link.

$$W_{loss} = 2^{-v_i} \cdot L \tag{1}$$

2) Influence of the delay: Due to path delay, the video packets might not reach the destination on time. Too late frame packets are discarded. While early frame packets are saved in the buffer and they wait for their time to decode and display. For this reason, higher priority is given to upper video layers, and upper layers are streamed over paths offering quicker delivery of these layers. Consequently, upper layers become available when the base layer reaches the client. To accomplish this behavior, the delay component W_{delay} of the edge weight is also weighted by the priority factor v_i for every i^{th} video layer. The delay component W_{delay} of the edge weight is given by Eq. 2, where D is the delay introduced by the link.

$$W_{delay} = (1 - 2^{-v_i}) \cdot D \tag{2}$$

3) Total edge weight: The total edge weight W of a link can be described as the weighted arithmetic mean of the loss component W_{loss} and the delay component W_{delay}, as defined in Eq. 3.

$$\begin{aligned} W &= (1-\beta) \cdot W_{loss} + \beta \cdot W_{delay} \\ &= (1-\beta) \cdot 2^{-v_i} \cdot L + \beta \cdot (1 - 2^{-v_i}) \cdot D \end{aligned} \tag{3}$$

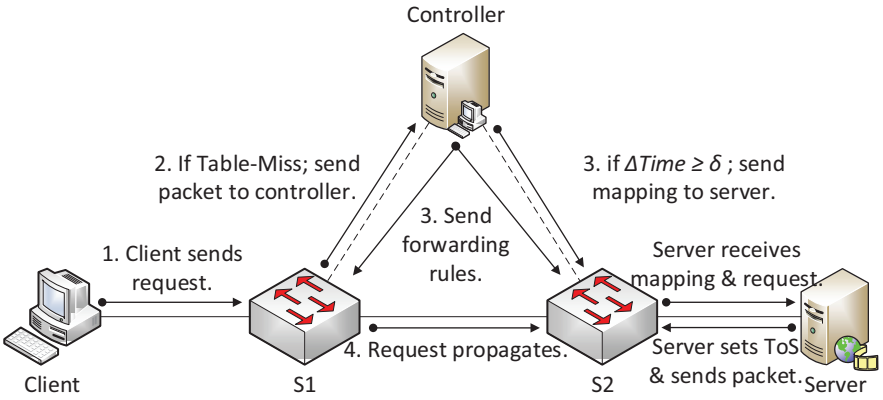

Figure 3: Conveying ToS mapping to a server

B. Influence of Flow-Timeouts

Every flow entry has a *HARD_TIMEOUT* and an *IDLE_TIMEOUT* associated with it. The switch notes the flow entry's arrival time, as it may require to remove the entry later. A non-zero *HARD_TIMEOUT* evicts a flow entry after the given number of seconds, irrespective of how many packets it has matched. A non-zero *IDLE_TIMEOUT* evicts a flow entry when it has not matched any packet in the given number of seconds [6]. In our proposed system, timeouts for flow entries are installed in the following manner:

1) Flow entries related to the video layers has only *IDLE_TIMEOUT* set.
2) Flow entries related to the background traffic have both *IDLE_TIMEOUT* and *HARD_TIMEOUT* set.

The existence of only *IDLE_TIMEOUT* for video related flow entries enables such entries not to be removed as long as the video transmission is active. It helps in avoiding the inevitable delay that incurs if *HARD_TIMEOUT* is set for an entry. Because removal of an entry causes a switch to re-consult the controller for an incoming packet that belongs to the expired flow entry.

The absence of *HARD_TIMEOUT* for video related entries causes switches to keep forwarding active video stream over the same path, despite a better path might be available as link's characteristics such as delay, etc., are not invariant. In our approach, when a client's request reaches the controller, it starts a timer for the client. Each time the timer for a client expires, the controller computes new paths for the video layers. If the new path differs from the previous path it sends corresponding forwarding rules to the switches. Otherwise, switches continue to apply previously installed rules to forward video packets.

C. Identifying Video Layers

When a client initiates a service request, it arrives at the switch to which the client is connected. Initially, the switch does not have a matching entry for this request. Hence, it sends a PACKET_IN message to the controller. The controller finds a path between the client and server and it instructs the corresponding switches to forward the request on an appropriate port. At the same time, the controller sends a mapping to the server which contains a layer identifier and corresponding IP Type-of-Service (ToS) field value for video layers. This mapping is called ToS mapping. While streaming, the server sets ToS field of the video packets according to the received ToS mapping. Figure 3 depicts the process of conveying ToS mapping to a server. Now, the controller can distinctly identify video layers by inspecting the ToS field value of incoming packets. Table I shows an illustrative example of the layer identifier and corresponding ToS field value for various video layers.

Table I: An illustrative example of ToS mapping

Video Layer	Layer Identifier	IP ToS Field Value
Base Layer	0	0x64
Enhancement Layer 1	1	0x6E
Enhancement Layer 2	2	0x78
Enhancement Layer 3	3	0x82
.	.	.
.	.	.
.	.	.

One of the major benefits of this approach is that the controller does not require to perform complex procedures, e.g., deep packet inspection of packet payload to identify video packets. Also, ToS field consumes only one byte in the packet header.

It is important to note that the server is not bound to receive the mapping. As a result, the server may miss it as well. One solution is to retransmit this mapping continuously. However, this would congest the controller-to-switch link. On the other side, several clients may send a request to the server at the same time. Sending mapping for each request will also congest the controller-to-switch link. In our proposed system, the controller records the time when it sends the mapping to a server. When a client's request comes to the controller, it checks when it last sent the mapping to the requested server. The controller sends the mapping to the requested server only if:

$$Time_{Current} - Time^{M}_{Last\_Sent} \geq \delta \tag{4}$$

M is the MAC address of requested server, δ defines the time duration between two consecutive transmissions of the mapping.

D. System Architecture

The implementation details of our approach are given in this section. Initially, the switches contain no entry in the flow tables. A packet from a host arrives at the switch to which the host is connected. The switch obtains the packet and sends a PACKET_IN message to the controller because initially its flow table has no matching entry. The controller determines an appropriate path for the packet and then sends forwarding rules to the corresponding forwarding elements on the computed path. The controller uses FLOW_MOD messages to send forwarding rules to the switches. Since a feature with a broad range of values can dominate the routing decision, the controller uses normalized values of link loss and delay while calculating routes. The controller interacts with the switches both in reactive and proactive manner. Whenever it receives a PACKET_IN message from a switch asking for rules, it reactively sends the forwarding rules to the switch. As a part of the proactive interaction, the controller periodically sends PORT_STATS and FLOW_STATS messages to the switches to acquire real-time traffic volume statistics. It also injects probe packets to calculate delay on the links, as described in [25]. The controller computes loss and available bandwidth on the links by utilizing the received traffic statistics.

IV. Evaluation

The details of experiment setup, performance metrics used and results are discussed in this section.

A. Experiment Setup

The evaluated network topology is illustrated in Figure 4. It has a POX [26] controller and four OpenFlow switches, each of the which is logically linked to the controller. The topology contains loops, providing multiple paths between server and client. Mininet [27, 28] is used to create the topology and perform the simulations. The video packets are sent by Real-time Transport Protocol (RTP) over User Datagram Protocol (UDP). Hence, we observe packet losses in every simulation. The value of δ and timer for each client is set to 10 seconds by the controller. Iperf [29, 30] client and server are used to generate Transmission Control Protocol (TCP) data stream which serves as the background traffic.

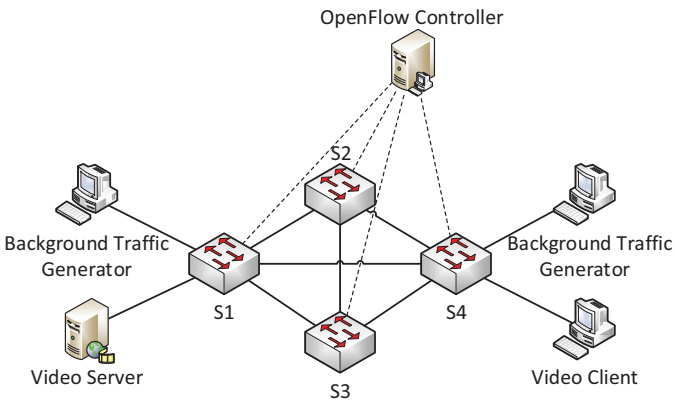

Figure 4: Evaluated network topology

The test video sequence used in the experiments is Foreman [31, 32]. It is in YUV CIF (352 x 288) format, and it consists of 300 frames. This video sequence is encoded by Joint Scalable Video Model (JSVM, version 9.19) [33] encoder with only temporal scalability enabled. The parameters of the resulting video are summarized in Table II.

Table II: Parameters of Foreman video

Layer	Resolution	Frame Rate (Fps)	Bit Rate (Kbps)	(DId, TId, QId)
0	352 x 288	7.5	514.10	(0,0,0)
1	352 x 288	15.0	548.70	(0,1,0)
2	352 x 288	30.0	588.10	(0,2,0)

The dependency id (DId) is used to define the spatial scalability inter-layer coding structure. The temporal id (TId) denotes the temporal scalability hierarchically. The quality id (QId) indicates the quality scalability structure.

B. Performance Metrics

We have compared the algorithms on the basis of average Peak Signal-to-Noise Ratio (PSNR), average frame loss rate and average throughput.

1) Average PSNR: PSNR provides an approximate measure of the quality as subjectively perceived by human observers. It is widely used because it has clear physical meanings. PSNR value for both luminance (Y-PSNR) and chrominance (U-PSNR and V-PSNR) components of the received video is calculated. A higher PSNR value corresponds to a better image quality.

2) Average Frame Loss Rate: A video frame is fragmented into multiple packets. The client cannot reconstruct the frame if any of these packets is lost. Therefore, we measure the percentage of lost frames instead of the percentage of lost packets as it expresses the perceived QoE more precisely.

3) Average Throughput: The video bit-rate received at the client is also measured. Improvement in the quality of the delivered video must not affect the rest of the network traffic. For this purpose average throughput for the Iperf client is also calculated.

C. Results

For a fair comparison and evaluation of the proposed algorithm, we performed ten runs of every algorithm. Averaged results are obtained and shown in the graphs. Figure 5, 6, 7 illustrate average PSNR value of Y, U and V components respectively for different algorithms. ELBA improves the Y, U and V components by 23.11%, 8.32%, 10.59% respectively when compared to shortest-path routing. Since the human eye is more sensitive to brightness (luminance) than color (chrominance), Y-PSNR typically holds more importance. Figure 8 shows the average frame loss rate for video traffic. The lower layer packets experience lesser losses because ELBA streams these packets over a less error-prone path. Successful delivery of lower layer packets leads to correct decoding of the frames. ELBA reduces frame loss rate by 39.56%. Figure 9 depicts the average bit-rate received for the video traffic while the average throughput for the background traffic is shown in Figure 10.

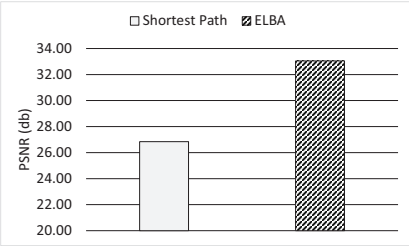

Figure 5: Average PSNR values for Y component

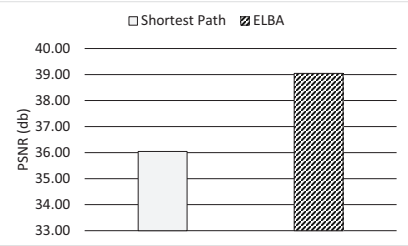

Figure 6: Average PSNR values for U component

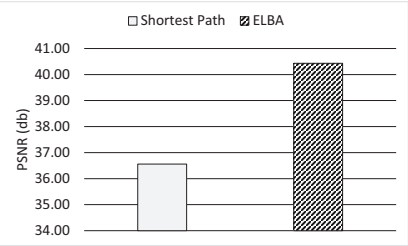

Figure 7: Average PSNR values for V component

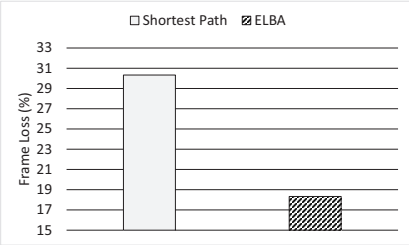

Figure 8: Average frame loss rate for video traffic

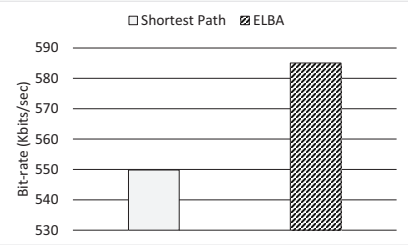

Figure 9: Average bit-rate received for video traffic

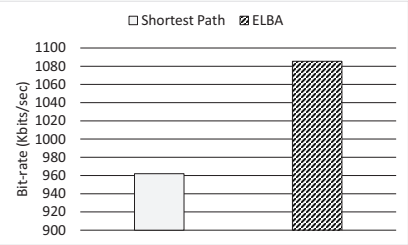

Figure 10: Average throughput for background traffic

As expected, shortest-path routing assigns the same path for each video layer as well as for the background traffic. In our approach, different video layers and the background traffic are streamed over their suitable paths. The received average video bit-rate is improved by 6.41% while the average throughput for the background traffic is improved by 12.83%.

Additionally, to illustrate how an end user perceives the difference in performance, we randomly chose and took snapshots of three consecutive frames from the received videos for each approach. Figure 11 illustrates a corresponding visual comparison in which frame number 94, 95 and 96 are snapshotted using a RAW video sequence player, i.e., PYUV [34]. By comparing the frames in Figure 11, it is evident that the video quality delivered using ELBA is better than conventional shortest-path routing approach.

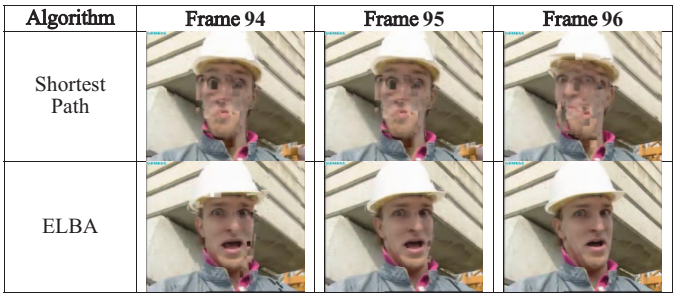

Figure 11: Visual comparison based on simulation

V. Conclusion and Future Work

In this paper, we have presented an algorithm for enhancing the quality of SVC-based scalable video streaming. SDN enables the forwarding devices to be updated dynamically, which allows us to stream different layers of SVC coded video over distinct network routes. As shown by the comparisons, the proposed algorithm significantly outperforms traditional shortest-path routing not only in terms of PSNR and frame loss rate, but it also delivers better throughput for both video traffic and background traffic.

In future, We shall explore how the inclusion of other metrics such as jitter and congestion improves the video quality. We also hope to present a comprehensive mathematical model and validate it using an exhaustive emulated environment. We shall also extend our approach to multi-domain networks containing more than one SDN controller.

VI. Acknowledgments

The work of Ankit Gangwal, Megha Gupta, Manoj Singh Gaur, and Vijay Laxmi was partially supported by Department of Electronics and Information Technology, Government of India project grant “Information Security Education and Awareness” (ISEA Phase- II) for Malaviya National Institute of Technology Jaipur as Resource Centre. Mauro Conti is supported by a Marie Curie Fellowship funded by the European Commission (agreement PCIG11-GA-2012-321980). This work is also partially supported by the EU TagItSmart! Project (agreement H2020-ICT30-2015-688061), the EU-India REACH Project (agreement ICI+/2014/342-896), the Italian MIUR-PRIN TENACE Project (agreement 20103P34XC), and by the projects “Tackling Mobile Malware with Innovative Machine Learning Techniques”, “Physical-Layer Security for Wireless Communication”, and “Content Centric Networking: Security and Privacy Issues” funded by the University of Padua.

References

[1] I. Cisco, “Cisco Visual Networking Index: Forecast and Methodology, 2014–2019,” *CISCO White paper*, 2015.

[2] T. Wiegand, G. J. Sullivan, G. Bjøntegaard, and A. Luthra, “Overview of the H. 264/AVC video coding standard,” *Circuits and Systems for Video Technology, IEEE Transactions on*, vol. 13, no. 7, pp. 560–576, 2003.

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

[4] Y. Jarraya, T. Madi, and M. Debbabi, "A survey and a layered taxonomy of software-defined networking," *Communications Surveys & Tutorials, IEEE*, vol. 16, no. 4, pp. 1955–1980, 2014.

[5] B. Nunes, M. Mendonca, X. N. Nguyen, K. Obraczka, T. Turletti *et al.*, "A survey of software-defined networking: Past, present, and future of programmable networks," *Communications Surveys & Tutorials, IEEE*, vol. 16, no. 3, pp. 1617–1634, 2014.

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

[7] Q. Ma and P. Steenkiste, "Supporting dynamic inter-class resource sharing: a multi-class QoS routing algorithm," in *INFOCOM'99. Eighteenth Annual Joint Conference of the IEEE Computer and Communications Societies. Proceedings. IEEE*, vol. 2. IEEE, 1999, pp. 649–660.

[8] K. Nahrstedt and S. Chen, "Coexistence of QoS and best-effort flows-routing and scheduling," in *Proceedings of 10th IEEE Tyrrhenian International Workshop on Digital Communications: Multimedia Communications*. Citeseer, 1998.

[9] Y. M. Hsiao, S. W. Yeh, J. S. Chen, and Y. S. Chu, "A design of bandwidth adaptive multimedia gateway for scalable video coding," in *Circuits and Systems (APCCAS), 2010 IEEE Asia Pacific Conference on*. IEEE, 2010, pp. 160–163.

[10] T. Schierl, C. Hellge, S. Mirta, K. Grüneberg, and T. Wiegand, "Using H. 264/AVC-based scalable video coding (SVC) for real time streaming in wireless IP networks," in *Circuits and Systems, 2007. ISCAS 2007. IEEE International Symposium on*. IEEE, 2007, pp. 3455–3458.

[11] M. S. Seddiki, M. Shahbaz, S. Donovan, S. Grover, M. Park, N. Feamster, and Y. Q. Song, "FlowQoS: QoS for the rest of us," in *Proceedings of the third workshop on Hot topics in software defined networking*. ACM, 2014, pp. 207–208.

[12] P. Georgopoulos, M. Broadbent, B. Plattner, and N. Race, "Cache as a service: Leveraging sdn to efficiently and transparently support video-on-demand on the last mile," in *Computer Communication and Networks (ICCCN), 2014 23rd International Conference on*. IEEE, 2014, pp. 1–9.

[13] A. Kassler, L. Skorin-Kapov, O. Dobrijevic, M. Matijasevic, and P. Dely, "Towards QoE-driven multimedia service negotiation and path optimization with software defined networking," in *Software, Telecommunications and Computer Networks (SoftCOM), 2012 20th International Conference on*. IEEE, 2012, pp. 1–5.

[14] P. McDonagh, C. Olariu, A. Hava, and C. Thorpe, "Enabling IPTV service assurance using OpenFlow," in *Advanced Information Networking and Applications Workshops (WAINA), 2013 27th International Conference on*. IEEE, 2013, pp. 1456–1460.

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

[16] I. Unanue, I. Urteaga, R. Husemann, J. Del Ser, V. Roesler, A. Rodríguez, and P. Sánchez, "A tutorial on H. 264/SVC scalable video coding and its tradeoff between quality, coding efficiency and performance," *Recent Advances on Video Coding*, vol. 13, 2011.

[17] S. Civanlar, M. Parlakisik, A. M. Tekalp, B. Gorkemli, B. Kaytaz, and E. Onem, "A qos-enabled openflow environment for scalable video streaming," in *GLOBECOM Workshops (GC Wkshps), 2010 IEEE*. IEEE, 2010, pp. 351–356.

[18] H. E. Egilmez, B. Gorkemli, S. Civanlar *et al.*, "Scalable video streaming over OpenFlow networks: An optimization framework for QoS routing," in *Image Processing (ICIP), 2011 18th IEEE International Conference on*. IEEE, 2011, pp. 2241–2244.

[19] H. E. Egilmez, S. T. Dane, K. T. Bagci, and A. M. Tekalp, "OpenQoS: An OpenFlow controller design for multimedia delivery with end-to-end Quality of Service over Software-Defined Networks," in *Signal & Information Processing Association Annual Summit and Conference (APSIPA ASC), 2012 Asia-Pacific*. IEEE, 2012, pp. 1–8.

[20] S. Laga, T. Van Cleemput, F. Van Raemdonck, F. Vanhoutte, N. Bouten, M. Claeys, and F. De Turck, "Optimizing scalable video delivery through OpenFlow layer-based routing," in *Network Operations and Management Symposium (NOMS), 2014 IEEE*. IEEE, 2014, pp. 1–4.

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

[22] T. Uzakgider, C. Cetinkaya, and M. Sayit, "Learning-based approach for layered adaptive video streaming over SDN," *Computer Networks*, vol. 92, pp. 357–368, 2015.

[23] E. W. Dijkstra, "A note on two problems in connexion with graphs," *Numerische mathematik*, vol. 1, no. 1, pp. 269–271, 1959.

[24] V. Kaibel and M. A. Peinhardt, *On the bottleneck shortest path problem*. Konrad-Zuse-Zentrum für Informationstechnik, 2006.

[25] V. N. Gourov, "Network Monitoring with Software Defined Networking: Towards OpenFlow network monitoring," Ph.D. dissertation, TU Delft, Delft University of Technology, 2013.

[26] POX. http://www.noxrepo.org/pox/about-pox/. Last accessed on Mar. 01, 2016.

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

[28] Mininet. http://mininet.org/. Last accessed on Mar. 01, 2016.

[29] A. Tirumala, F. Qin, J. Dugan, J. Ferguson, and K. Gibbs, "Iperf: The TCP/UDP bandwidth measurement tool," *http://dast. nlanr. net/Projects*, 2005.

[30] Iperf. https://iperf.fr/. Last accessed on Mar. 01, 2016.

[31] P. Seeling and M. Reisslein, "Video transport evaluation with H. 264 video traces," *Communications Surveys & Tutorials, IEEE*, vol. 14, no. 4, pp. 1142–1165, 2012.

[32] Video Trace Library. http://trace.eas.asu.edu/. Last accessed on Mar. 01, 2016.

[33] Team, Joint Video, "H. 264/SVC Reference Software (JSVM 9.19) and Manual.(2011)," *Retrieved October*, vol. 8, p. 2014, 2011.

[34] PYUV. http://dsplab.diei.unipg.it/software/pyuv_raw_video_sequence_player. Last accessed on Mar. 01, 2016.