In-Network Video Quality Adaption using Packet Trimming at the Edge

Stuart Clayman
Dept. of Electronic Engineering,
University College London,
London, UK
Email: s.clayman@ucl.ac.uk

Mustafa Tüker, Emre Karakış, Müge Sayıt
International Computer Institute,
Ege University,
Izmir, Turkey
Email: {tukermustafa, emrekarakis}@gmail.com,
muge.sayit@ege.edu.tr

***Abstract*—This paper describes the effects of running in-network quality adaption by trimming the packets of layered video streams at the edge. The video stream is transmitted using the BPP transport protocol, which is like UDP, but has been designed to be both amenable to trimming and to provide low-latency and high reliability. The traffic adaption uses the Packet Wash process of BPP on the transmitted Scalable Video Coding (SVC) video streams as they pass through a network function which is BPP-aware and embedded at the edge. Our previous work has either demonstrated the use of SDN controllers to directly implement Packet Wash, or the use of a network function in the core of the network to do the same task. This paper presents the first attempt to deploy and evaluate such a process in the edge. We compare the performance of transmitting video using BPP and the Packet Wash trimming, against alternative transmission schemes, namely TCP, UDP, and HTTP Adaptive Streaming (HAS). The results demonstrate that providing traffic engineering using in-network quality adaption using packet trimming, provides high quality at the receiver.**

***Index Terms*—Edge Computing, High-speed Packet Processors, Packet Trimming, Traffic Engineering, SVC Video**

I. Introduction

Recent emerging technologies in networking such as the use of softwarized and virtualized network functions and edge computing, together with new protocols, provides a flexible infrastructure which can be tailored to the various requirements specific to Internet applications. Multimedia applications, being one of the most popular applications on the Internet, have become demanding of resources. This is amplified with the growth in user expectations and the characteristics of challenging applications such as remote surgery, AR/VR, and mobile broadcasting. Therefore, although current video streaming applications and standards achieve a good performance today, those future applications will continue to force network and service operators to evolve towards applications providing lower latency, higher reliability, and more adaptivity.

The high traffic volume created by video streaming applications can cause an increase in packet losses, which results in increased latency due to the retransmissions. Although there are a number of techniques for addressing this in the server and the client, *Packet Trimming* is a promising technique that can be used to reduce the number of packets lost [1]. Packet Trimming is a relatively new idea which relies on very fast hardware to adjust the size of a packet during its transmission over the network [2] [3]. When the network becomes limited, the packets are trimmed according to the bottleneck link capacity rather then dropping the whole packet.

Currently, none of the standard protocols support packet trimming directly. BPP is a protocol that can utilize trimming in its design [4]. BPP has been designed as one of the protocols used for the low latency and high reliability applications in future networks. From its introduction in 2018 [4], there are a number of studies related to BPP, which show its usage. The use of the Packet Wash process, where chunks in packets are trimmed, has been shown to reduce latency in [2].

In this paper, we propose an architecture, which is based on Packet Trimming and the BPP protocol to provide low latency with high Quality of Experience (QoE) for video streaming applications. We consider a scenario, where there are clients having different resolutions and rendering capabilities, hence, the quality of the video should be adapted. Although, video streaming, using HTTP Adaptive Video Streaming (HAS), is the most prevalent and successful application in which the clients adapt the video quality on the basis of observed network parameters, in this study, we implement in-network video quality adaptation with the packet trimming approach.

In general, Wide Area Networks (WAN) have the capacity to transfer video with high quality, however, the bottleneck links are at the edge where the clients connect to the network. As each client is different, their network attributes are different, and the rendering capabilities of each device can be different, a streaming system has many aspects to factor in. By considering these, in our architecture the server always transfers the video at the highest quality, and we rely on the edge to address the client aspects. We utilize the Packet Wash facilities in BPP to implement packet trimming, and we virtualize the BPP-aware network functions at the edge. When the packets arrive to the edge, the BPP-aware function handles them by implementing packet trimming based on the characteristics of each client, and then sends the potentially trimmed packets to the client. The video transmitted is encoded by using a layered codec, namely H.264 Scalable Video Coding (SVC) which makes video packet trimming suitable because of its characteristics that allows quality adaptation by extracting layers.

No one had previously tested sending media streams using packet trimming, but we designed and built an implementation

of such a system, and provided some preliminary evaluations in our previous work [5] and [6]. Those initial results showed that in-network video quality adaptation can be a promising technique that can meet the requirements of emerging and future video streaming applications. In those studies, we used an SDN controller that implements the BPP functions. However, these studies also showed that the implementation architecture has a remarkable impact on QoE. Processing in the core network creates more load at the center when handling multiple streams, it is also far from the client, and it is harder to deal with the various client differences. Hence, we propose a new architecture that provides higher QoE, is able spread the processing load, and is closer to the clients. The contributions of this paper are threefold. First, we show the potential advantages of in-network quality adaptation by comparing it to HAS. The next contribution is an architecture that uses the edge and virtualized BPP functions for delivery. We compare various protocols, and provide experimental results showing the QoE obtained, and we also discuss the issues caused by implementing the packet trimming process directly in an ONOS SDN controller, showing that deep packet inspection and packet updates in ONOS is highly CPU intensive and is not scalable. This new architecture is compatible with 5G/6G and NFV and provides scalability and with optimized results.

The paper is structured as follows: Background is next, with a description of the BPP Process in Section III, a comparison of In-Network Quality Adaption versus HAS in Section IV, a description of doing BPP processing at the edge in Section V, and an evaluation of this in Section VI. The conclusions and further work are in Section VII.

II. Background

Layered video codecs such as Scalable Video Coding (SVC) enables video sequences with various qualities from one encoded video file [7]. After the video is captured, the video frames are encoded with various parameter settings, which produces a number of quality alternatives within the video, by taking advantage of the similarities between the different versions of the same frame. Using SVC video has been shown to have a beneficial impact on video transmission [8].

The transmission of video is usually done in one of 2 ways: (i) as discrete packets using RTP over UDP, which is unreliable and can have loss at the receiver, or (ii) as data streams using HTTP over TCP, which is reliable, but can have delay / latency at the receiver. With UDP, the receiving application has to deal with packet loss in the network, but the application does have direct control over requests for resends, if they are needed. Using UDP works well when *low latency* is important. With TCP, the receiver application is responsible for dealing with the delay, which is commonly done by using buffering techniques. However, as the application is presented with a stream of bytes, it has no control over requesting resends which is directly implemented in the TCP stack of the kernel. Using TCP works well for video playback, where there is no interactivity.

RTP is a protocol [9] devised to carry media streams with a UDP transport. RTP packets have relative timestamps, and it supports both Sender reports that have a mapping relative to NTP timestamp, which can be used this for syncing, and Receiver reports which present packet loss/jitter. RTSP is a presentation-layer protocol that allows end-users to interact with media servers via pause and play capabilities. RTSP is a stateful protocol used for media delivery [10], using RTP. WebRTC is a combination of standards, protocols, and JavaScript APIs that enables Real-Time Communications via a web browser [11], with a latency of sub-500 milliseconds. WebRTC uses SRTP (Secure Real-time Transport Protocol) to ensure that the exchanged data is secure and authenticated.

The HTTP protocol sits on TCP [12], and is common for transmitting video, such as those used by CDNs, include HTTP Adaptive Video Streaming (HAS) which is today's de-facto streaming technology. Dynamic Adaptive Streaming over HTTP (DASH) [13] is a standard developed by MPEG, for providing interoperability between the participants of HAS deployments. The DASH server is designed to send segments of video, of a few seconds length, at the requested quality. The client can dynamically adapt the quality of the video being sent by requesting segments of a different quality, and placing those video segments into the decoder for viewing. To accommodate various bandwidths, the source video is encoded multiple times with a range of qualities. The higher the quality, the higher the data rate, the bigger the file, and the more bandwidth used.

BPP was designed to be used for low latency and high reliability applications [4]. It has been evaluated in a number of studies to determine its effectiveness. In [14], combining BPP with TSN (Time Sensitive Networking) was used to overcome the limitations of TSN's scalability and complexity issues. In [15], BPP was used for computation offloading in Mobile Edge Networks. Neither of those papers addressed multimedia over BPP. Although BPP was designed for media transmission, our previous work [5] and [6] was the first actual implementation of video streaming using BPP, and for determining the effects of sending video over BPP. In [16] we demonstrated that in-network adaption can be provided when using SVC combined with the BPP protocol. A more detailed description of the techniques and mechanisms used for carrying the streams of SVC video from servers to clients, using the BPP Packet Wash mechanism is presented in our paper [17].

Work on Packet Trimming has become of interest as hardware has become fast enough to do in-transit packet updates. Handley et al. considered trimming for the Data Center [1]. They define NDP, a novel data-center transport that achieves near-optimal completion times for short transfers and high flow throughput in a wide range of scenarios. With NDP, the switches trim the packets to just headers and then priority forward the headers. To improve round trip times across the network, in [3] the authors suggest that trimming the whole payload and only keeping the header, can only work well in Data Centers where the dropped payload may be retransmitted fast enough to the node that dropped the payload. Such a data center approach does not generalize to full WANs. Selectively

trimming the packet, rather than the whole payload, offers a less drastic mechanism that would work in the wide area network. For enhancing end-to-end transport, a new transport protocol is defined QUCO [18], which reacts to congestion by selectively dropping parts of a payload packet (combined with mitigation mechanisms to handle the loss of part of the payload). Their scheme reduces the variation in the number of packets going through the network, and has less delay and less delay variations than TCP, with a resulting reduction in jitter. Very recent work is [19], which investigates how trimming can be implemented in existing programmable switches which were not specifically designed for trimming. The implementation using P4 on the Tofino switch ASIC is presented.

Edge Computing is considered an extension of cloud computing, whereby additional computational, data handling, and networking resources are placed closer to the end systems. As a consequence, the processes for data processing, networking, data management, and storage can occur between the end systems and the cloud servers, not just at the centralized cloud servers. Edge Computing can be extremely useful for low-latency applications, as well as applications that generate an enormous amount of data that cannot be practically transferred to cloud servers in real-time, due to proximity to the clients, or bandwidth or time limitations [20]. In recent times, centralized applications have evolved towards service-oriented architectures and microservices, with small independent and loosely coupled systems that deal with a very specific tasks being virtualized and executed autonomously, especially at the edge.

III. BPP Processing

To execute the required BPP processing of packets, BPP enabled network nodes are necessary. The server is responsible for constructing the packets, by creating the chunks and by setting the *significance value* of each chunk.Our paper [17] explains this process in detail. Fig. 1 shows the steps of the Packet Wash process, whereby chunks can be trimmed if the total amount of data transferred in a specific time period exceeds the available bandwidth. 1 Packets are constructed with a BPP header and a number of chunks, and transmitted across the network. 2 When the packet arrives at a network node, it is sent for processing. Each programmable network node enumerates the number of bits transferred within each time period, using the size of the packets. If the value exceeds the available bandwidth, then it determines that trimming should occur. 3 The node checks each packet to evaluate which chunks should be trimming from the packet, according to the available bandwidth measurements. Chunks are trimmed one at a time, by considering the *significance value* field of each chunk, such that the size of a packet should be reduced to below the specified limit. Chunks whose *significance value* is lower than a threshold are not trimmed in any circumstances, and so the packet size may not be reduced as much as desired. This means later packets will need more trimming. 4 The packet, modified or not, is sent back to the switch, and 5 forwarded onwards. The client is responsible for collecting the chunks from the packets, and reconstructing a valid data stream, as shown in step 3 of Fig. 2.

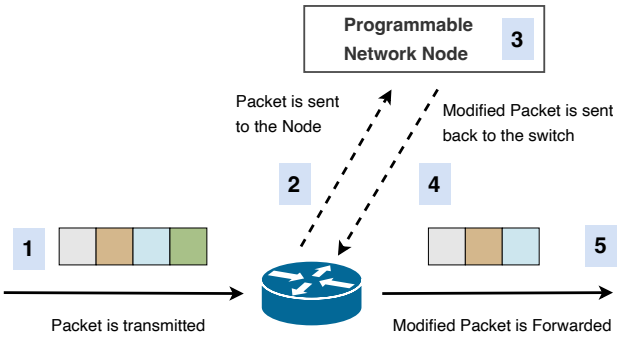


Fig. 1: BPP Processing

The effects of Packet Wash trimming ensures that the client receives a continuous stream of packets, and so always has some meaningful data to process. This approach of using the bandwidth is clearly different from UDP, where packets are dropped when there is congestion. Although there is no more usable bandwidth when sending with BPP, our previous work has shown that it can be utilized and managed more effectively.

We presented the idea of in-network quality adaptation in our work [16]. That paper showed that it was a promising approach when compared to other both TCP and UDP. In [21] we showed the full effects of Packet Wash on SVC video in limited bandwidth environments. HTTP Adaptive Video Streaming (HAS), being the one of the most popular video streaming applications, provides an efficient client driven adaptation when network conditions change. HAS clients adapt the quality on the basis of both the observed and the internal parameters, hence minimizing the negative impacts of network condition changes on QoE. In this paper, we focus on the potential advantages of in-network quality adaptation, at the edge, over quality adaptation at the client.

IV. In-Network Quality Adaption versus HAS

To show the potential advantages of in-network quality adaption we compare it to HAS in order to determine its effectiveness. Only by doing this we can determine if such an approach has the benefits we are looking for. The first approach we considered, in [5] and [6], utilized an ONOS controller to implement the BPP packet wash process.

The video used in all the experiments is *Big Buck Bunny*. The bitrate distribution for video qualities are slightly different for HAS and other protocols, due to encoding aspects. For the HAS experiments, we used the packetized video for HAS systems given in [22]. The bitrates of the qualities of the packetized encoded video used with HAS are 1 Mbps, 2 Mbps, and 4.2 Mbps. In the other experiments, using BPP, UDP, and TCP, the bitrates of the video layers are 1.1 Mbps, 1.9 Mbps, and 5.6 Mbps. However, these bitrates are increased after the packetization. We decreased the bandwidth values in accordance with the bitrate of the packetized video, in order to provide a fair comparison in HAS experiments.

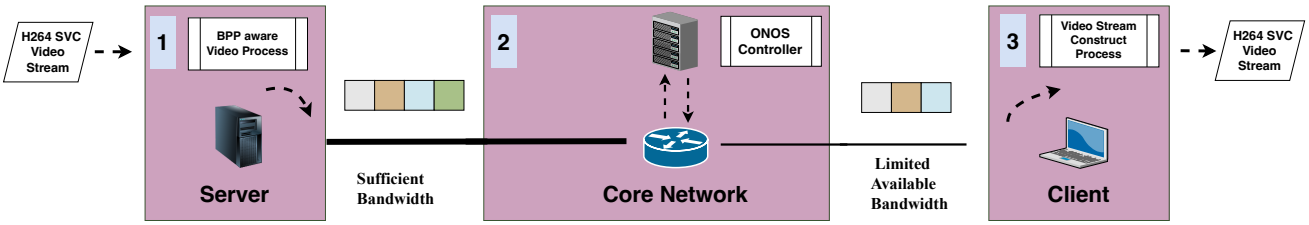
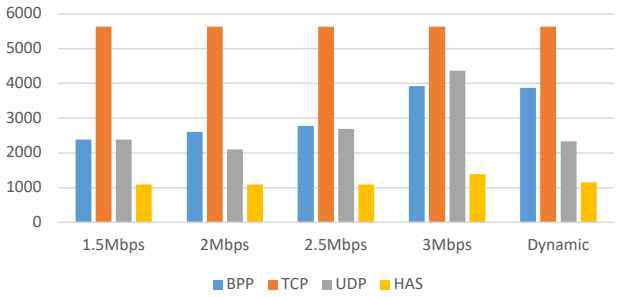
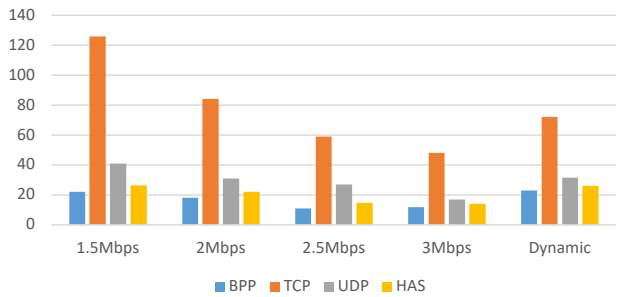


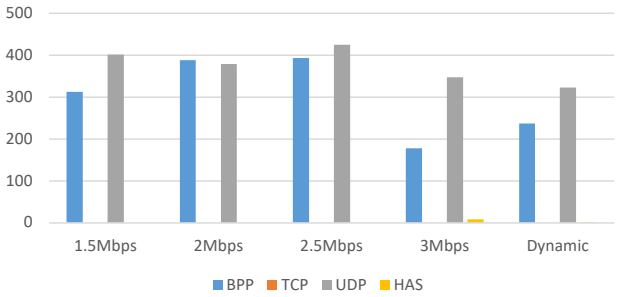
Fig. 2: ONOS Controller for BPP Processing



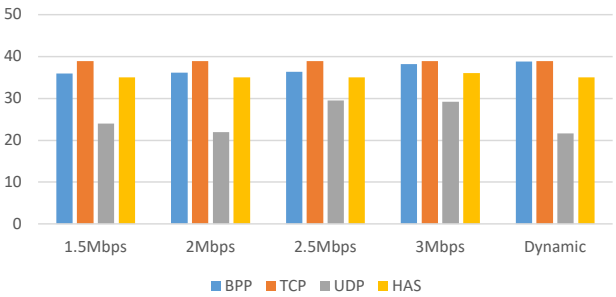
(a) Average Video Bitrate
Higher is Better



(b) Total Duration Of Pauses
Lower is Better



(c) Quality Changes
Lower is Better



(d) PSNR
Higher is Better

Fig. 3: Quality Parameters using ONOS Controller for BPP Processing

A. *On the Performance of In-Network Quality Adaption*

We conducted a set of experiments using an ONOS controller, based on the architecture in Fig. 2, with a number of network conditions, on a Mininet environment [23]. We compared the quality adaption with BPP against adaption with HAS, as well as having regular transmission with UDP and TCP, where there is no quality adaption. In these experiments, the bandwidth of the path between the core network and the client is *limited* and set to different values, namely 1.5 Mbps, 2 Mbps, 2.5 Mbps, and 3 Mbps. There is also one set of experiments with dynamic network conditions, where the bandwidth changes between 1 Mbps and 3 Mbps over time.

In Fig. 3, the different QoE parameters observed by different protocols are presented. In Fig. 3a, the average bitrate of the played video on the client's side are given. In terms of this QoE parameter, TCP outperforms other approach since TCP client always get the video at the highest quality. However, this behaviour has significant costs, with a high level of duration of pauses, which is the one of the attributes that has a huge negative impact on QoE, as seen in Fig. 3b. Due to its design, the HAS client wisely applies a conservative approach to keep the duration of pauses at minimum, where the clients always request the video at the lowest quality. As the TCP and HAS implementations do not change the quality over time, there is almost no observable quality changes with these experiments, in Fig 3c, whereas BPP and UDP have many changes. In Fig. 3d, the PSNR values observed for each protocol are given. We see that TCP has the highest PSNR value since the server sends video at the highest quality during the whole session. BPP and HAS values for PSNR are similar, where the PSNR values obtained with BPP is slightly higher than HAS.

Although the different QoE parameters observed in the experiments provide some basic information about the nominal perceived quality on the client side, an *overall QoE value* gives a better idea about the general perceived quality. To calculate this overall QoE value, a linear function is defined

$$QoE_1^K = \alpha . \sum_{k=1}^{K} Q_k - \beta . \sum_{k=1}^{K-1} |l_{k+1} - l_k| - \gamma . \sum_{k=1}^{K} dp_k \quad (1)$$

$$dp_k = \begin{cases} ts_k - p_k & ts_k \geq p_k \\ 0 & otherwise \end{cases} \quad (2)$$

based on one devised in [24]. The importance of different QoE parameters is specified by setting different weights to the different terms, based on their effects on perceived quality.

The *QoE value* formula is given in Equation 1 and it calculates the overall QoE for the segments numbered between k and K. In the formula, α, β, and γ are the weights used for the QoE parameters. Q_k represents the video quality in terms of PSNR or video bitrate for the k^{th} segment. l_k is the layer of the k^{th} segment, therefore the second term in the formula given the quality change level between two consecutive segments. dp_k represents the duration of pauses experienced during the playout of the k^{th} segment, and it is calculated by the function given in Equation 2. In that function, ts_k and p_k are the timestamp value of the received time and the playout time of the k^{th} segment, respectively.

The overall QoE values calculated for this set of experiments are given in Table I. In the QoE calculation, the PSNR value is used as the Q_k values in equation 1. The weights in the equation, for α, β, and γ, are 2, 0.01, and 0.5, respectively. These numbers are selected based on the positive / negative effects of the QoE parameters. The research done on perceived quality [25] showed that the bitrate is the most important parameter that affects QoE, and that users prefer quality changes over a duration of pauses. The table shows that the highest overall QoE values are observed with BPP in fixed bandwidth experiments, however, HAS values that are very close to the BPP values. With dynamic bandwidth conditions, we observe that BPP outperforms other approaches, giving an insight into how in-network quality adaption can be beneficial compared to client-based adaptation.

	Bandwidth				
	1.5 Mbps	*2 Mbps*	*2.5 Mbps*	*3 Mbps*	*Dynamic*
BPP	57.7	59.5	63.4	68.8	63.7
TCP	14.8	35.8	48.3	53.8	41.8
UDP	23.5	24.7	41.2	46.4	24.1
HAS	56.9	59	62.7	64.9	56.9

TABLE I: Overall QoE values for different transmission schemes and varying bandwidths

B. On the Effects of Implementing BPP in the Controller

The performance results show that BPP adapts the quality in an efficient way, as there is a lower total duration of pauses and a higher received average bitrate, when compared to HAS. However, HAS managed to keep the number of quality switches to a minimum, when compared to BPP.

The experiments given in the previous section were conducted with one server, one client, and one OpenFlow enabled switch. The controller sets flow rules in the switches, between the server and client, at the beginning of the streaming session, to forward the packets. We observe that if there is just one client on the network, the SDN controller can manage the BPP processing for each packet of the stream. However, if the number of clients increases, problems arise and the controller starts consuming a high level of CPU resource. Additionally, sending a packet to the controller from a switch causes an increase in the end-to-end delay due to an extended transmission delay between the controller and the switch.

We assume that the network operator and the video streaming company are in co-operation, so the controller gets knowledge about the highest video bitrate from the server. This information helps the controller to determine if the bandwidth is enough to transfer the video with the highest quality. If the available bandwidth of a link is too low to send the video with the highest quality, then the packets should be trimmed at that point. During the session, it periodically measures available bandwidth. In order to limit the computational complexity on ONOS and reduce the end-to-end delay, we developed an approach which would only send packets to the controller when a BPP operation was needed, rather than sending every packet. To support this, the controller *removes* the flow rule related to the video stream from the switch, if the bandwidth becomes limited. When a packet is received by the switch *and* there are no flow rules, it sends a message to the controller to ask for the flow information related to the video stream. As the response, the controller sends the trimmed packet, coupled with the output port information.

We implemented this approach and performed several experiments on Mininet. However, the experiments show that the scalability of this approach is quite limited since it requires Deep Packet Inspection (DPI) operations that trims the payload and headers. Given the number of issues with processing BPP streams in the controller, including having too much load with so few streams, we investigated another potential solution. We proceeded to a new approach and evaluation, which is to provide in-network quality adaption by implementing BPP functions as a virtual network function. We replaced the ONOS controller with one virtual router implementating a BPP-aware function, located between the server and the client. In our paper [21] we show how well that approach works.

V. BPP Processing at the Edge

In this paper, we generalized the idea of the virtualized BPP function implementation and propose a system for larger networks with more clients connected, where a virtualized BPP function is used for the BPP operations.

A. Virtualized BPP Function

In general, the bottleneck links are those links that connect clients to the network. Therefore, in the proposed system, the virtualized BPP-aware network functions are installed at the edge routers. In our system, the edge network is an SDN

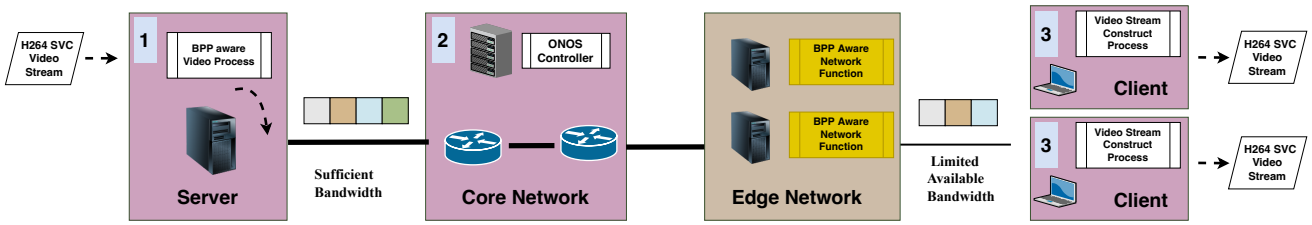
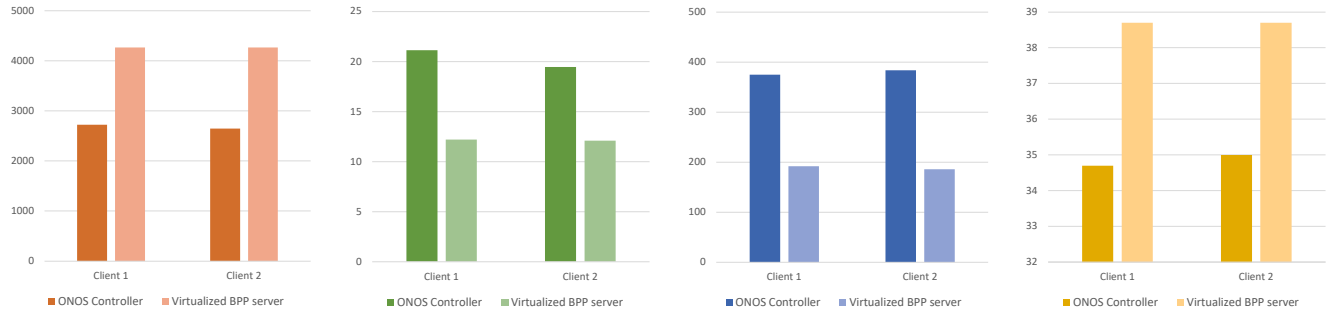


Fig. 4: Edge Network Deployment of VNFs



(a) Average Video Bitrate
Higher is Better

(b) Total Duration Of Pauses
Lower is Better

(c) Quality Changes
Lower is Better

(d) PSNR
Higher is Better

Fig. 5: Comparing BPP Process Performance
ONOS Controller versus Virtualized BPP Server at the Edge

domain where the controller is responsible for managing the network. It periodically measures the available bandwidth of links that clients connected to the network. This available bandwidth information is fed to the BPP function.

In the system, shown in Fig. 4, the server sends the video at the highest quality, considering there are many clients having different characteristics, such as different bandwidth conditions, device resolutions, and rendering capabilities. The packets, which carry all the layers of the video, are transferred through the network until they arrive to the edge. At the edge, the packets are sent to the BPP function, where the packets are trimmed according to the characteristics of each client, such as the current link bandwidth between the edge and the client, and then sends the potentially trimmed packets to the client.

When the virtualized BPP function receives a packet, it determines whether the packet should be trimmed and if it is, how many chunks should be removed from the packet. The virtualized BPP function uses the available bandwidth information of each client based on the information received from the controller. Packet trimming is done by using a simple algorithm. For each packet that the function receives, it checks if the number of bytes sent in the current second is proportional to the available bandwidth. If the number of bytes sent is greater than the available bandwidth, it determines that bytes should be washed. The trim operation is done by removing the chunks based on their *significance value*, as described in Section III step 3. As the size of the chunks that can be trimmed may be different to the number of bytes that need to be trimmed, it can take a number of packets to be trimmed in order to match the correct bandwidth level. The packets, modified or not, are distributed to the clients.

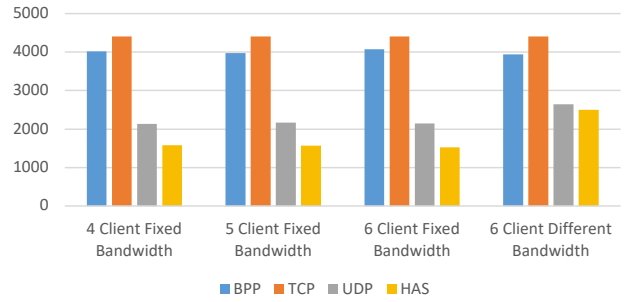
B. On the Performance of Implementing BPP at the Edge

Here we present a comparison of two different approaches to process packet trimming. For this purpose, the video is streamed between the server and two clients, and the QoE parameters which are measured on the client side are collected.

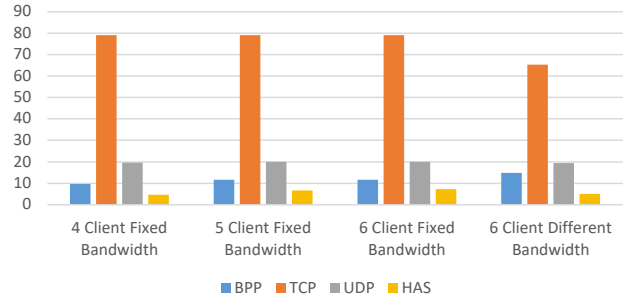
These results for Client 1 and Client 2 are in Fig. 5. The bars labeled "ONOS Controller" represent the results that are observed when using an ONOS controller which has a module running the BPP process. On the bars labeled as "Virtualized BPP server", the BPP processing is done by a virtualized function at the edge.

In these experiments, the 2 clients are connected to the network over different edge links, and the available bandwidth on those links is 2.5 Mbps. When the packet trimming operation is done by the virtualized BPP function, all of the QoE parameters have better values, compared to the experiments with the ONOS controller, as seen in Fig. 5. In addition to that, although the clients received the video at a higher quality, the duration of pauses observed with the virtualized BPP server is still less than ONOS controller due to the latency added by the BPP process on the controller as seen in Fig. 5b.

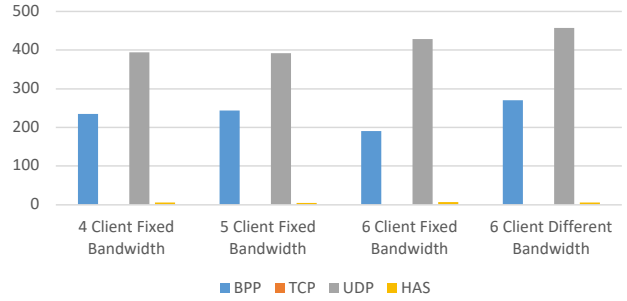
Overall, we see that the QoE obtained by trimming packets at the edge is better than using the ONOS controller, as utilizing the edge virtualized BPP functions, the QoE parameters are higher than the those with the ONOS controller.



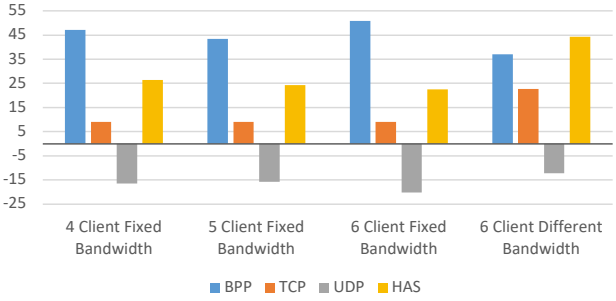
(a) Average Video Bitrate
Higher is Better



(b) Total Duration Of Pauses
Lower is Better



(c) Quality Changes
Lower is Better



(d) Overall QoE
Higher is Better

Fig. 6: Quality Parameters using Virtualized BPP Servers and Multiple Clients

VI. Performance Evaluations

In this section, we give the performance evaluation of the virtualized BPP function which is compared with the other protocols, UDP, TCP, and HAS. In BPP, UDP, and TCP experiments, the server sends the video at the highest bitrate.

A. Experimental Setup

The video used here is encoded with slightly different parameters than that of used previously. The bitrate of the base, first enhancement, and second enhancement layers is 0.9 Mbps, 1.9 Mbps, and 4.4 Mbps, respectively. When the overhead of packetization using BPP is added, we obtain similar bitrates of the encoded video used in HAS experiments.

In the experiments, where there are 4, 5, and 6 clients connected to the server, those are conducted with a fixed bandwidth value of 2.5 Mbps. In the experiments for different bandwidth, there are 6 clients connected to the network, having bandwidth changes between 2.5 Mbps and 4 Mbps. Each experiment is reconfigured for HAS, UDP, and TCP, using the same networks conditions. HAS is inherently adaptive, but there is no quality adaptation in the experiments with UDP and TCP. The initial buffering time is set to 600 ms for all approaches, to provide a transmission with low latency. The performance results observed in these experiments follows.

B. Comparative Performance Evaluation with Multiple Clients

The QoE parameters are collected and averaged in this set of experiments for each protocol. In Fig. 6a, the average received video bitrate values are presented for each protocol. The clients using TCP play the video with the highest quality, since there is no quality adaptation. The clients with UDP do not adapt quality, so the bitrate values of the UDP clients are lower than TCP due to the lost packets. The clients using BPP play the video with a higher quality compared to UDP and HAS clients. In Fig. 6b, we see that the HAS clients manage to keep the duration of pauses lower than all other protocols due to their policy of keeping duration of pauses at minimum, based on their buffer fullness. The number of quality changes, in Fig. 6c, is high with BPP since it does not consider this aspect during the packet wash decision making.

In addition, the *Overall QoE value* is calculated by using Equation 1. In the enumeration of the *QoE value*, the average received video bitrate is used as the Q_k parameter. The α parameter, which shows the importance of the video bitrate is set to 0.02. This is lower than the α value used in section IV-A because the bitrate unit is in Kbps, which is higher than the other parameters, namely the number of the quality switches and the duration of pauses. We also increase the penalty for the number of quality switches and the duration of pauses, and use the weights 1 and 0.1, for β and γ, respectively. When we examine the overall QoE values, Fig. 6d, we see that BPP outperforms other approaches with the fixed bandwidth experiments. In the experiments conducted with different bandwidth values, the highest overall QoE value is observed with HAS. The reason for that is this set of

experiments has clients with higher bandwidth connections, so those HAS clients can play the video with higher bitrates.

VII. Conclusions

In this paper we have shown the advantages of having in-network quality adaptation by presenting a number of performance results and comparing it to HAS. The comparative experiments show that in-network video quality adaptation is a promising approach that can meet the requirements of future video streaming applications. The results show that BPP adapts the quality in an efficient way, such that lower total duration of pauses and higher average received bitrate can be obtained when compared to HAS. However, HAS managed to keep the number of quality switches minimized, compared to BPP.

An architecture that utilizes virtualized BPP functions at the edge, for video delivery, has been presented. We showed the use of an ONOS controller as a solution to implement in-network quality adaptation, but on the other hand, it does add a huge burden to the controller since it also has the responsibilities to manage the network. The experimental results have shown that implementing in-network quality adaptation at the edge, and by using a virtualized BPP function, provides scalability and an improvement in QoE.

We compared a number of protocols in this paper, and demonstrated good performance via the experimental results, which has shown that the QoE obtained by trimming packets at the edge is better than using an ONOS controller. In the experiments utilizing the virtualized BPP functions, the average video bitrate is higher than those experiments utilizing the ONOS controller. The insights observed from this study show that in-network video quality adaption might provide enhanced QoE. Overall, the QoE value could be even higher if more refined approaches, which consider the number of quality changes, are developed. Nonetheless, doing quality adaptation at the client side will always have the advantage of receiving information about internal parameters, where one of the most important among them being the buffer level.

For future work, we will consider a number of improvements. We will adapt the algorithm that trims the chunks from the packets. Currently, it only pays attention to the available bandwidth and the bandwidth used, and does not consider the number of quality changes. This number is quite high in the experiments, but for better QoE values, and for perceptual reasons, it is ideal to reduce the number of quality changes for smoother delivery. We will also investigate the use of packet trimming using hardware systems, if these become available, in order to provide higher throughput.

Acknowledgment

Dr S. Clayman is partially supported by Huawei Technologies Co., Ltd. This work is funded by TUBITAK Electric, Electronic and Informatics Research Group (EEEAG) under grant 121E373.

References

[1] M. Handley, C. Raiciu, A. Agache, A. Voinescu, A. Moore, G. Antichi, and M. Wójcik, "Re-Architecting Datacenter Networks and Stacks for Low Latency and High Performance," in *ACM SIGCOMM*, Aug 2017, pp. 29–42.

[2] L. Dong and R. Li, "In-packet network coding for effective packet wash and packet enrichment," in *2019 IEEE Globecom Workshops (GC Wkshps)*, 2019, pp. 1–6.

[3] C. Westphal, K. Makhijani, and R. Li, "Packet Trimming to Reduce Buffer Sizes and Improve Round-Trip Times - Extended Abstract," in *Buffer Sizing Workshop (BS'19)*. New York, NY: ACM, 2019.

[4] R. Li, A. Clemm, U. Chunduri, L. Dong, and K. Makhijani, "A new framework and protocol for future networking applications," in *NEAT 2018 - Proc. of ACM Workshop on Networking for Emerging Applications and Technologies*, August 2018.

[5] S. Clayman, M. Tuker, H. Arasan, and M. Sayıt, "The Future of Media Streaming Systems: Transferring Video over New IP," in *IEEE 22nd Intl. Conf. on High Performance Switching and Routing (HPSR)*, Paris, June 2021.

[6] ——, "Managing Video Processing and Delivery using Big Packet Protocol with SDN Controllers," in *IEEE Conf. on Network Softwarization – Netsoft*, Tokyo, July 2021.

[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, October 2007.

[8] S. Yang, Z. Wei, and L. Ling, "Multimedia Communication and Scalable Video Coding," in *Fourth Intl Conf on Intelligent Computation Technology and Automation*, Shenzhen, China, 2011, pp. 616–619.

[9] "RTP: A Transport Protocol for Real-Time Applications." [Online]. Available: https://tools.ietf.org/html/rfc3550

[10] "Real Time Streaming Protocol (RTSP)." [Online]. Available: https://tools.ietf.org/html/rfc2326

[11] "Chromium Blog." [Online]. Available: https://blog.chromium.org/2020/05/celebrating-10-years-of-webm-and-webrtc.html

[12] "Hypertext Transfer Protocol Version 2 (HTTP/2)," RFC 7540. [Online]. Available: https://httpwg.org/specs/rfc7540.html

[13] "MPEG-DASH: Dynamic Adaptive Streaming over HTTP." [Online]. Available: https://mpeg.chiariglione.org/standards/mpeg-dash

[14] K. Makhijani, R. Li, and H. E. Boukary, "Using Big Packet Protocol Framework to Support Low Latency based Large Scale Networks," in *ICNS 2019*, Athens, 2019.

[15] L. Dong and R. Li, "Distributed Mechanism for Computation Offloading Task Routing in Mobile Edge Cloud Network," in *International Conference on Computing, Networking and Communications (ICNC)*, Honolulu, HI, USA, 2019, pp. 630–636.

[16] S. Clayman and M. Sayit, "In-Network Scalable Video Adaption Using Big Packet Protocol," in *Proceedings of the 12th ACM Multimedia Systems Conference*, ser. MMSys '21. ACM, 2021, p. 363–368.

[17] S. Clayman and M. Sayıt, "Low Latency Low Loss Media Delivery Utilizing In-Network Packet Wash," *Journal of Network and Systems Management*, vol. 31, no. 1, p. 29, 2023.

[18] A. Albalawi, H. Yousefi, C. Westphal, K. Makhijani, and J. Garcia-Luna-Aceves, "Enhancing End-to-End Transport with Packet Trimming," in *GLOBECOM 2020 - 2020 IEEE Global Comms Conf*, 2020, pp. 1–7.

[19] A. Popa, D. Dumitrescu, M. Handley, G. Nikolaidis, J. Lee, and C. Raiciu, "Implementing Packet Trimming Support in Hardware," 2022. [Online]. Available: https://arxiv.org/abs/2207.04967

[20] K. Cao, Y. Liu, G. Meng, and Q. Sun, "An Overview on Edge Computing Research," *IEEE Access*, vol. 8, pp. 85 714–85 728, 2020.

[21] S. Clayman and M. Sayıt, "The Effects of Packet Wash on SVC Video in Limited Bandwidth Environments," in *IEEE 23rd Intl. Conf. on High Performance Switching and Routing (HPSR)*, Jiangsu, China, June 2022.

[22] S. Lederer, C. Müller, and C. Timmerer, "Dynamic adaptive streaming over http dataset," in *Proceedings of the 3rd Multimedia Systems Conference*, ser. MMSys '12. New York, NY, USA: Association for Computing Machinery, 2012, p. 89–94.

[23] 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*. Association for Computing Machinery, 2010.

[24] X. Yin, A. Jindal, V. Sekar, and B. Sinopoli, "A control-theoretic approach for dynamic adaptive video streaming over http," *SIGCOMM Comput. Commun. Rev.*, vol. 45, no. 4, p. 325–338, aug 2015. [Online]. Available: https://doi.org/10.1145/2829988.2787486

[25] M. Seufert, S. Egger, M. Slanina, T. Zinner, T. Hoßfeld, and P. Tran-Gia, "A Survey on Quality of Experience of HTTP Adaptive Streaming," *IEEE Comms Surveys Tutorials*, vol. 17, no. 1, pp. 469–492, 2015.