The Effects of Packet Wash on SVC Video in Limited Bandwidth Environments

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

Müge Sayıt
International Computer Institute,
Ege University,
Izmir, Turkey
Email: muge.sayit@ege.edu.tr

***Abstract*—This paper describes the effects of the Packet Wash process on the transmission of layered SVC video streams. We show how the packet size is adapted when using a number of different packing strategies, that map the video data into the BPP packets, and discuss the relationship between the packing strategies on the sender side and the chunk removal in the washing process. We demonstrate how the packing strategy causes different impacts on the number of and the sizes of the washed chunks. As the bandwidth reduces, more of the chunks in a packet get washed away. Although the receiver gets packets that are much smaller than those transmitted by the sender, it is still able to play video with a high QoE as zero packets are dropped. This traffic engineering enables a direct implementation of an *in-network video adaption scheme*. The experimental evaluation highlights that the effects of Packet Wash become more obvious in environments where there is limited bandwidth.**

***Index Terms*—Future Networks, High-speed Packet Processors, Packet Wash, Traffic Engineering, SVC Video**

I. Introduction

In this paper we describe the effects of the Packet Wash process, introduced in [1], on the transmission of layered SVC video streams. Our experimental evaluations highlight that the effects of Packet Wash become more obvious in environments where there is limited bandwidth. In particular, as the bandwidth reduces, more of the chunks in a packet can be washed away. Consequently, the receiver gets many packets that are much smaller than those transmitted by the sender.

The Packet Wash mechanism, introduced for the Big Packet Protocol (BPP), was one of a number of protocols devised for the requirements of future network architectures and new high-precision services. BPP aimed to provided a framework for the needs of those high precision services, providing specific service level guarantees. More recently, a new protocol called New Internet Protocol (New IP) was proposed to overcome some of the limitations of traditional networks and to extend BPP [2]. New IP extends the capabilities and the header of BPP, but still requires support from network elements. Our work and implementation uses the defined packet structure of BPP, but as New IP is an enhancement to BPP, the same techniques and observations apply to New IP as well.

Scalable Video Coding (SVC) is an approach for creating video sequences with a number of different video qualities being held in a single video file [3]. The frames of the video are encoded with varying parameters, so that the resulting video contains enhanced quality layers of the original video stream. Layered video coding is predicated of the similarities between the encoded layers of the same frame, as each higher layer is applied over a lower layer. The similarities between frames over time are also factored into the encoding.

A mechanism has been created by which video frames from the different layers of the SVC video are multiplexed and packed into BPP packets. The structure of a packet meets the BPP spec, with a header and a number of data chunks. The header holds the size of each of the chunks, plus the offset in the packet. This data is complemented with the significance value of each chunk, and the relevant BPP commands for the network node. High-speed packet processors which implement BPP, or New IP, can do on-the-fly traffic engineering by adapting the packet contents, during the Packet Wash process. This process eliminates video chunks of low priority, but keeps critical video chunks, and by having specific labelling of video data at the sender, this approach ensures that there is always at least one chunk that can be delivered to the client, even if other chunks are removed in each packet.

Using these concepts and techniques, we have successfully built a working proof-of-concept system that implements these mechanisms. The system and the results clearly demonstrate that when transmitting SVC video with the BPP protocol, and having network nodes that support the Packet Wash mechanism, we enable a direct implementation of an *in-network video adaption scheme* which is able to change the video characteristics during transmission. Although the video frames at the server are not always the same frames as the video at the client, due to the washing process, we show that the video plays with good QoE attributes. Our preliminary analysis shows that the Packet Wash feature of BPP helps to cope with bandwidth changes and hence helps to increase the performance of video streaming applications.

In this paper, we analyse how the packet size is adapted when using a number of different packing strategies, that map the video data into the BPP packets. The contributions of this paper can be listed as: *(i)* a discussion on the relation between the chunk removal approach in the BPP washing process and the packing strategies on the sender side, in SVC

video steaming; *(ii)* demonstrating how the packing strategy causes different impacts on the number and the sizes of washed chunks; and *(iii)* an evaluation of these strategies in terms of the goodput obtained at the client's side. To the best of our knowledge, this is the first study that shows the joint effects of a packing strategy and packet wash approach, with respect to the various bandwidth conditions.

II. Background

It is common for a large number of video delivery systems to use DASH [4]. DASH, which is Dynamic Adaptive Streaming over HTTP, is an MPEG standard devised to ensure system interoperability, and uses the benefits of HTTP caching and HTTP transport for video delivery. DASH relies on a pull model, where the client makes continual requests for a large number of very small video files, which may be cached somewhere in the network. Furthermore, as HTTP sits on TCP as its underlying transport protocol, this provides a reliable mechanism for delivery, with no loss seen in the receiving application. However, in limited bandwidth environments packets are lost in the network during transmission, and they are resent, which does cause latency.

Other video streaming approaches use a push model, where video is sent from a source to a receiver. These often use UDP for transport, to provide a faster communication method, sometimes with RTP layered on top [5]. With UDP, any loss in the network has to be dealt with by the application, as there is no in-built mechanisms for re-transmission, nor are there algorithms for adjusting the transmission rate, such as slow start and the congestion control of TCP. The QUIC protocol, which is like HTTP layered on top of UDP, was introduced to improve the responsiveness of web services, by overcoming some of the issues of using TCP [6]. There is a belief that QUIC will therefore inherently be good for transmitting video. In both [7] and [8] it was found that video over QUIC has some drawbacks, even compared to having a TCP transport.

BPP is a transport protocol, just like UDP and TCP. As such it has its own attributes, and can be used to carry various types of traffic, and can have other protocols layered on top of it. In [9] the concept of Qualitative Communication was proposed to reduce the granularity of packet drops due to network congestion, from packet level to chunk level. The desired effect of this is to improve latency that a user experiences and to reduce the number of packets being retransmitted, compared to reliable transport schemes such as TCP. Among the various commands that are supported by both BPP and New IP, we utilise the Packet Wash command, which allows shrinking of the packets during transmission [1]. This feature helps to change the packet size dynamically, as it facilitates the removal of some chunks within the packet, rather than dropping the whole packet when the bandwidth is limited. Transferring some data to the client side is generally better than no data, or delayed data, when the network is congested [1].

The concept of carrying SVC video with BPP as initially presented in [10]. We have taken these concepts and have built a working proof-of-concept system that implements these ideas. In our previous work, we presented the full structure of a BPP packet used for SVC video, plus the definitions of the main blocks in [11]. That paper discusses the enhancements to BPP needed for SVC, describes the use of BPP for carrying video from servers to clients, plus the extensions needed to support SVC encoded video. We presented the design and use of an SDN controller to implement the Packet Wash of transmitted SVC video [12]. The SDN controller used information on the characteristics of the video coding together with real-time network conditions to process and transmit the video with a focus on a high Quality of Experience at the client. In [13] we directly compare the performance of network transport protocols, by comparing BPP vs UDP vs TCP QoE metrics, rather than focussing on application layer framing, such as RTP, which is independent of the transport. That work shows that in-network adaption can be provided using our approach. Here, our analysis shows that the Packet Wash feature helps to cope with bandwidth changes and hence helps to increase the performance of video streaming applications.

III. Packet Washing Video Streams

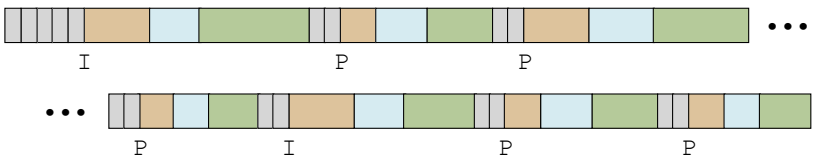
We now present how SVC video is: (i) read from a video stream; (ii) how the content is packed into BPP chunks ready for transmission over the network; (iii) how Packet Wash can be used to reduce the packet size under limited bandwith conditions; and (iv) how the receiver handles the situation when chunks are missing, if they have been washed away.

1) SVC Video Structure: In an H264 video the byte stream is structured as a sequence of segments, called the Network Abstraction Layer (NAL). A NAL may contain video data, and is labelled VCL, or may contain meta data, and is labelled NONVCL. These NONVCL NALs are generally small (shown in grey in Fig. 1a), whereas the video VCL NALs can range in size, being small for low-res small scale video, to large for high-res large scale 4K, and now 8K, videos.

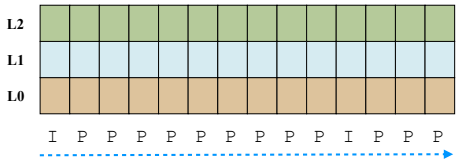
Overall the structure of a SVC video is similar to a *normal* H264 file, where we see 2 kinds of NAL, with a single VCL NAL for each frame type: I, P or B. For SVC videos, we see multiple NALs for a frame. In our work, we used a 3 layer encoding, which results in 3 NALs per layer. The video is stored as a byte stream, with the NALs laid out sequentially in the stream, as in Fig. 1a. The layers can be viewed conceptually as that in Fig. 1b. Layer 0, which provides the base layer information, and Layer 1 and Layer 2, which are applied on top to provide enhancements to the base layer.

2) Content Chunks Packing Strategies: Here we show the different packing strategies utilized when taking SVC video from that data stream, and putting it into packets.

The NONVCL NALs which tend to be small, can easily be placed into a packet, and these get sent as a single BPP chunk. There are 3 VCL NALs for each of the video frames in our video stream, as seen in Fig. 1a. In general, the frame size and the NAL size is much bigger than a standard 1500 byte packet, so the sequence of NALs has to be collected for each frame, and then mapped into a number of BPP packets. This task utilizes three phases for the VCLs:



(a) VCL NALs (brown, blue, green) and NONVCL NALs (grey) are in the stream sequentially



(b) VCL NALs over time conceptual view – layer 0 (base layer), layer 1, and layer 2

Fig. 1: Video NALs Structure – Sequential in stream & Conceptual

1) Collecting and enumerating the sequence of NALs from the video stream. In this phase an H264 stream processor collects 3 VCL NALs from the input;
2) Splitting the 3 collected NALs for each frame, and *packing* then into a list of intermediate objects, which we call ChunkInfo objects; and
3) Taking the ChunkInfo objects and converting them into the actual BPP packets, for transmission.

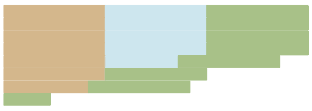
There are currently 4 *packing* strategies that we use:

- *Even split* – This strategy splits the data evenly into equal size chunks. When having 3 NALs we allocate the same number of bytes per chunk.
- *Dynamic split* – This strategy determines how many chunks need to hold data, based on the input amounts from the NALs. It firstly tries an even split, and if these are filled, it ends. If not, it reallocates space by increasing the allocation to the NALs with remaining data.
- *In Order* – This strategy creates chunks in the order they come from the NALs. Data from the NAL will fill the packet, until the last chunk. All other chunks are 0.
- *Fully Packed* – This strategy is similar to In Order, except that with the last chunk, the remaining data is allocated to the next NAL.

If we consider the effects of these packing strategies on the structure of the BPP packets and the sizes of the chunks, the data in the tables of Fig. 2 shows this. We use a 1500 byte packet, having a content size of 1472 bytes, and then allow for the BPP header and BPP meta-data before allocating data to the chunks. The original NAL sizes for 3 VCL NALs are as follows: Layer 0: 3232 bytes, Layer 1: 2232 bytes, and Layer 2: 3527 bytes. Fig. 2 shows the result of the *packing* strategies that maps the NALs into the chunks in each packet.

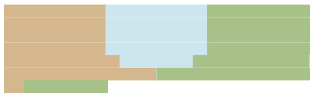
3) Network Node Packet Wash Process: As described, Packet Wash enables the elimination of a number of chunks, during the transmission of the packet. It changes the packet size dynamically, rather than dropping the whole packet. The chunks that have a high priority are kept, whilst low-priority chunks can be dropped when there is congestion or when the bandwidth is limited. For SVC video all of layer 0 and all 3 layers of I frames are the most important. We set the highest priority to those elements. Layer 1 facets, have the next priority level, and layer 2 has the lowest priority level. Even though the packet payload can be smaller than the data sent from the server, this is better than receiving no data at all [1]. This

L0	L1	L2
473	473	473
473	473	473
473	473	473
473	473	473
473	340	473
473	0	473
394	0	473
0	0	216



(a) Even Split Strategy

L0	L1	L2
473	473	473
473	473	473
473	473	473
473	473	473
539	340	539
709	0	709
92	0	387



(b) Dynamic Strategy

L0	L1	L2
1419	0	0
1419	0	0
394	0	0
0	1419	0
0	813	0
0	0	1419
0	0	1419
0	0	689

(c) In Order Strategy

L0	L1	L2
1419	0	0
1419	0	0
394	1025	0
0	1207	212
0	0	1419
0	0	1419
0	0	477

(d) Fully Packed Strategy

Packets of 1500 bytes, with chunk sizes. One row per packet. The graphic has a visual representation of each packet. Original NAL sizes (in bytes) – *L0*: 3232, *L1*: 2232, *L2*: 3527

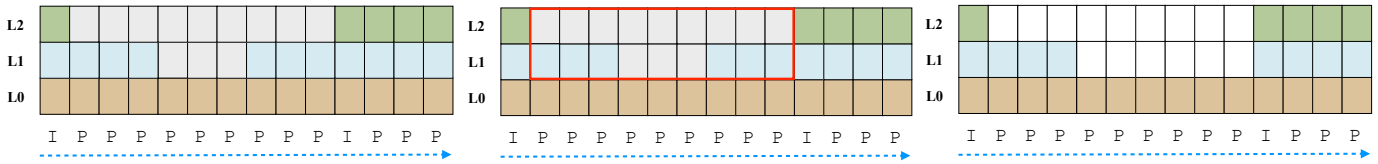
Fig. 2: Different packing strategies have different chunks

allows data to arrive at the receiver while still providing some usable information, as opposed to seeing a dropout or loss.

An impact of using these different *packing* strategies is that the packets are not a fixed size. Some networking schemes do not use strategies such as this, and will attempt to fill the 1500 bytes of a packet fully, by reading from the stream continuously until the stream ends. Our approach is based more on the *structure* of the video, and as seen in Fig. 2, the packet size does vary. The average packet size in bytes for each strategy with the selected *Foreman* video, is as follows:

Even Split	836	Dynamic	1276
In Order	976	Fully Packed	1276

This can have an impact on the Packet Wash, as when it eval-



(a) Washed NALs (in grey) (b) Dependency analysis region (red box) (c) Lost NALs not written (in white)

Fig. 3: Client-side NAL Processing – Handling Missing Chunks

uates estimated throughput figures for the video transmission, the results can be less accurate when compared with fixed size packets, resulting in a chunk elimination process that is slightly less accurate. However, by packing the video based on its structure and having varying packet sizes, we have shown it provides seamless throughput at the receiver [13]. Of particular note is that this chunk reduction process helps to prevent the dropping of whole packets, particularly when the bandwidth is limited, as it reduces the size of the packets, thus allowing more packets to flow down a connection with limited bandwidth, thus Packet Wash provides an efficient technique for managing streaming video.

4) Receiver Handling of Missing Chunks: When the server sends the packets, the network node might implement the Packet Wash process, depending on the available bandwidth value, and some of the NAL chunks within the packets are removed. These packets are further processed when they arrive to the client side, as the receiver checks the received NALs are consistent, and might remove more NALs according to the layer dependencies. We have designed a process that deals with the situation where chunks from the video have been removed. This process ensures that the resulting video stream is valid, and does not have any "holes", which would make the decoder stall or fail.

The receiver collects all the packets for a sequence of 3 VCLs, and unpacks them into the individual NALs. If a chunk is missing, it means that the NAL which contains the chunk cannot be rebuilt in a valid way, and so the received chunks of that NAL cannot be usefully written to the output video, because the decoder lacks reliable mechanisms to deal with damaged and inconsistent NALs. This first step ensures there is a method for when there are chunks missing, to ensure consistency in the NALs. This is shown in Fig. 3a, where the NALs with missing chunks, drawn in grey, are not candidates to be written out. Although some of those NALs are valid, there are still some dependencies between the layers and consecutive frames that need consideration.

For the second step, these NALs are analysed by considering the layer dependencies. We do not need to put out Layer 1 or Layer 2 NALs if the preceding frames have not been reconstructed, particularly if they never made it from the I frame. Fig 3b shows the region of the video, between one I frame and another I frame, where we do this dependency analysis and cleanup. During the analysis we observe that some NALs which have lost their dependent NAL are not needed and can be *cleaned* away. Thus, if a NAL does not have its previous dependent frame, then it can never be decoded correctly, and so we do not need it. This is shown in Fig. 3c, where the white layers represent the washed and cleaned NALs that do not get written. As such, the output video contains a valid and stable bitstream to feed to the video decoder, which helps considerably during playback, and provides a a stable video stream for the experiments presented next.

These 4 strategies allow us to send SVC video over the network, and accomodate any loss due to the Packet Wash process, and provide a valid video stream for the decoder.

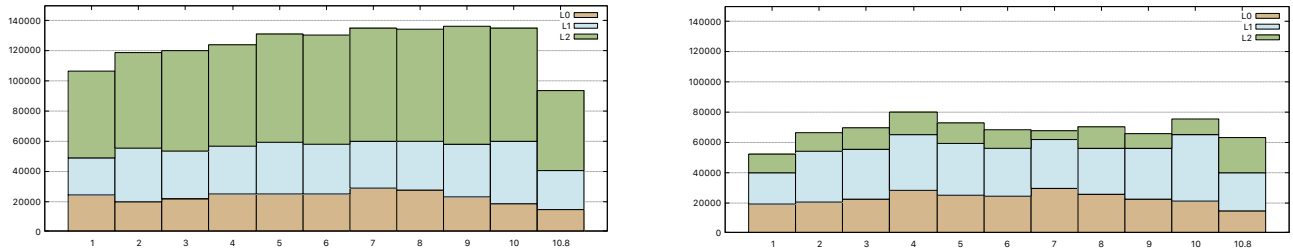
IV. Performance Evaluations

In order to observe the performance of different packing strategies with respect to the different bandwidth values, we conducted several experiments and examined the number of washed NALs and lost NALs non-written out.

1) Evaluation Setup: For these experiments, there is one server and one client, which are connected to each other via a BPP-enabled virtual router. We use the *Foreman* video [14], where the file is encoded with one base layer and two enhancement layers. The bitrates are 288 Kbps, 488 Kbps, and 1.094 Mbps for the base and two enhancement layers, respectively. The tests are run with a set of bandwidth values between 0.5 Mbps and 1.4 Mbps. The experiments are repeated, where the server sends packets with each type of packing strategies.

2) The Effects of Wash Process on the Packets: When the BPP-enabled router receives a packet, if the current bandwith is not enough to send the whole packet, it performs the packet wash process, and some chunks, with varying sizes, can be trimmed during this process. The number of bytes transmitted and bytes received when the bandwidth is 0.8 Mbps, are shown in Fig. 4. As seen from Fig. 4b, the client received all the bytes from layer 0 and layer 1, for this bandwidth. However, since the bandwidth 0.8 Mbps is not enough to send the video with the highest bitrate (1.094 Mbps), the router trimmed some chunks from layer 2 during transmission. Further content from layer 2 can be also be removed after this process, due to the dependency analysis that is done on the client side.

As mentioned in the previous section, the NALs can be either removed by the router, via the packet wash process, or cleaned and not written by the client, according to layer dependency analysis process. Fig. 5 shows the washed NALs and not written NALs for each type of packing strategy. There are less washed and not written NALs with *Dynamic* and *Fully Packed* strategies when compared to the *In Order* strategy. As shown in Fig. 2, the chunk distribution within the packets



(a) Bytes transmitted over time (secs) – per layer

(b) Bytes received over time (secs) – per layer

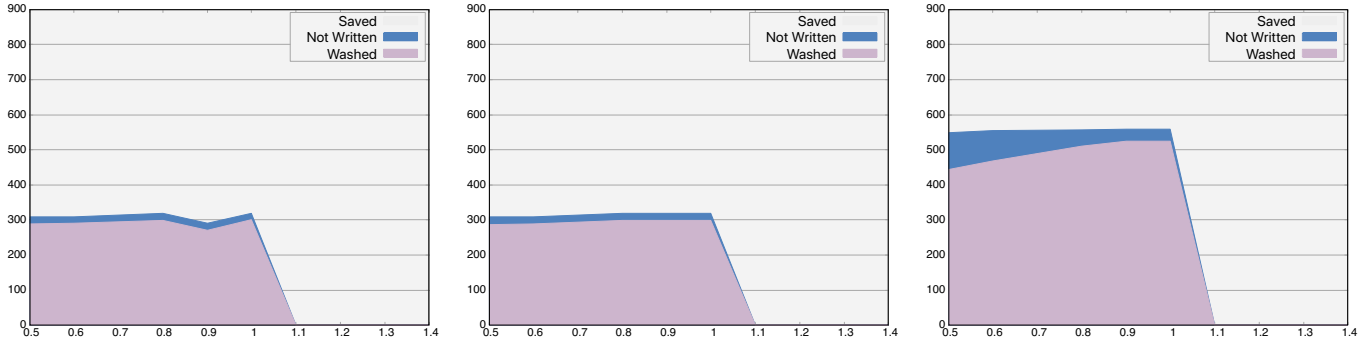
Fig. 4: Transmitted vs. received video data, highlighting in-network dropped chunks. Bandwith: 0.8 Mbps.

for an example frame is presented. There are different chunks belonging to each of the 3 layers with the *Dynamic* and *Fully Packed* strategies, thereby the router can trim chunks of the higher layers when it needs to shrink the packet. On the other hand, we see that with with the *In Order* strategy there are only chunks of one layer. With this strategy, the router cannot implement the prioritization among the chunks within a packet as there is only one chunk, and as a consequence, if the chunk is trimmed, the whole content of the packet is removed.

At the client side, more of the NALs are cleaned and not written to the output, due to the layer dependency analysis process when using *In Order*, compared to the other strategies. We see this effect by observing the size of the blue area in Fig. 5c. For those bandwidth values less than 0.8 Mbps, the number of NALs that are not written out increases as the bandwidth decreases. However, the number of NALs not written, barely changes for the limited bandwidth settings with the other strategies. ■ *Our first observation is that the packing strategy does not just affect the washed chunks during transmission, it can also affect the not written chunks on the client side, if there is a dependency between chunks. Hence, chunk dependencies should be also factored in when using different packing strategies.*

The received NALs on the client side for the *Dynamic*, *Fully Packed*, and *In Order* strategies, with respect to the different bandwidth values are given in Fig. 6a, Fig. 6b, and Fig. 6c respectively. In these graphs, the NALs from each layer, received at the client, shows the effects of washed and not written NALs to the application. All base layer NALs are received by the client for each strategy, therefore, the client is always able to play the video seamlessly. However, the quality of the video differs for each of the strategies due to NAL loss.

Similar performance in terms of received number of layers is observed with *Dynamic* and *Fully Packed* strategies, although the chunk distribution within the packets are not similar with these strategies. While most of the first enhancement layer NALs are received with *Dynamic* and *Fully Packed* strategies, we see that those NALs are dropped with *In Order* strategy, when the bandwidth is less than 1 Mbps in Fig. 6c. The reason for this can be seen by examining the example chunk distribution within the packets, for different packing strategies shown in Fig. 2. As seen in Fig. 2c, there is only one chunk in each packet with *In Order* strategy. Hence, if the router decides to trim a packet carrying those chunks, it has to trim all the content. We observe that even if there is enough capacity to send the base and the first enhancement layer for the available bandwidth values higher than 488 Kbps, layer 1 packets are not received with *In Order* strategy. However, with *Fully Packed* and *Dynamic* strategies, the router has the option to choose whether to trim layer 1 or layer 2 chunks in some packets. This flexibility has huge impact on the client side as seen in Fig. 6. Even *Dynamic* strategy provides this flexibility, although a small fraction of packets carry chunks having different significance values. ■ *Our second observation is that when there are chunks with different significance values within some of the packets, then the performance can be better.*

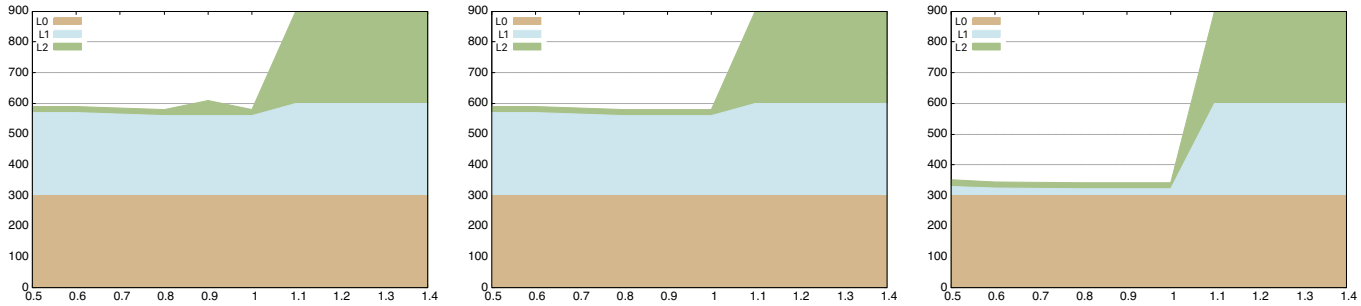


(a) Dynamic Strategy

(b) Fully Packed Strategy

(c) In Order Strategy

Fig. 5: Washed NALs and Not Written NALs



(a) Dynamic Strategy (b) Fully Packed Strategy (c) In Order Strategy

Fig. 6: Received NALs at the Client

V. Conclusions

BPP provides in-network adaptation of the packet size during transmission thanks to its Packet Wash mechanism. This capability and flexibility can be customized by considering the application needs. The Packet Wash mechanism enables the trimming of packets, rather than dropping the whole packet when the bandwidth is limited. With a scalable video codec, the video frames can be encoded with more than one layer, where each layer additionally increases the quality. Conversely, the layers can easily be individually extracted when a quality decrease is necessary. Therefore, the nature of scalable video codecs is well suited to BPP transmission.

As well as the chunk trimming strategy, chunk significance values have an impact on the performance at the client side. Also, a BPP aware packing strategy implemented on the server side has an importance. In this work, various packing strategies are proposed so that the NALs produced by H.264 SVC streams can be multiplexed into the packets by following different rules. The study shows that these different packing strategies affect the performance in terms of the number of received layers for SVC transmission over BPP.

The performance measurements show that if the packets are constructed with a different number of chunk types, the BPP enabled router has the flexibility on the decision of which chunks should be trimmed. These characterization and traffic engineering methods lead to higher performance. It is possible to use different chunk removal approaches for the packet wash process, by considering the application specific characteristics. In order to improve the performance further, the dependencies among chunks should be also considered when choosing a packing strategy and a chunk removal method.

It is expected that the emerging networking architectures such as 6G will customize network resources according to the applications needs. BPP / NewIP is a promising protocol that can run over future network architectures and enable partially reliable communication. With such characteristics, low-latency and high-precision requirements of applications can be met. As future work, we plan to expand our system to provide low-latency communication for multi-party clients, with different bandwidths, by utilizing the advantages of SDN and run the Packet Wash functions on an enhanced SDN controller.

Acknowledgment

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

References

[1] R. Li *et al.*, "A Framework for Qualitative Communications Using Big Packet Protocol," in *NEAT 2019: Proc. of ACM Workshop on Networking for Emerging Applications and Technologies*, August 2019, pp. 22–28.

[2] R. Li, K. Makhijani, and L. Dong, "New IP: A Data Packet Framework to Evolve the Internet : Invited Paper," in *2020 IEEE 21st Intl. Conf. on High Performance Switching and Routing (HPSR)*, 2020, pp. 1–8.

[3] 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.

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

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

[6] M. Bishop, "Hypertext Transfer Protocol Version 3 (HTTP/3)," Internet Engineering Task Force, Internet-Draft draft-ietf-quic-http-34, Feb. 2021. [Online]. Available: https://datatracker.ietf.org/doc/html/draft-ietf-quic-http-34

[7] M. Seufert, R. Schatz, N. Wehner, and P. Casas, "QUICker or not? -an Empirical Analysis of QUIC vs TCP for Video Streaming QoE Provisioning," in *22nd Conference on Innovation in Clouds, Internet and Networks and Workshops (ICIN)*, 2019, pp. 7–12.

[8] M. Palmer, T. Krüger, B. Chandrasekaran, and A. Feldmann, "The QUIC Fix for Optimal Video Streaming," in *Proceedings of the Workshop on the Evolution, Performance, and Interoperability of QUIC*, ser. EPIQ'18. ACM, 2018, p. 43–49.

[9] L. Dong, K. Makhijani, and R. Li, "Qualitative Communication Via Network Coding and New IP: Invited Paper," in *2020 IEEE 21st International Conference on High Performance Switching and Routing (HPSR)*, 2020, pp. 1–5.

[10] S. Clayman, "The Inter-Dependence of Network Transport and Application Behaviour," in *ITU Network 2030*, Geneva, October 2019. [Online]. Available: https://www.itu.int/en/ITU-T/Workshops-and-Seminars/2019101416/Documents/Stuart_Clayman_Presentation.pdf

[11] 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.

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

[13] 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.

[14] "Foreman, Xiph.org Video Test Media." [Online]. Available: https://media.xiph.org/video/derf/