Journal of Network and Systems Management (2023) 31:29
https://doi.org/10.1007/s10922-022-09712-1

Low Latency Low Loss Media Delivery Utilizing In-Network Packet Wash

Stuart Clayman[1] · Müge Sayıt[2]

Received: 3 June 2022 / Revised: 11 October 2022 / Accepted: 18 December 2022 /
Published online: 23 January 2023

Abstract

This paper presents new techniques and mechanisms for carrying streams of layered video using Scalable Video Coding (SVC) from servers to clients, utilizing the Packet Wash mechanism which is part of the Big Packet Protocol (BPP). BPP was designed to handle the transfer of packets for high-bandwidth, low-latency applications, aiming to overcome a number of issues current networks have with high precision services. One of the most important advantages of BPP is that it allows the dynamic adaption of packets during transmission. BPP uses Packet Wash to reduce the payload, and the size of a packet by eliminating specific chunks. For video, this means cutting out specific segments of the transferred video, rather than dropping packets, as happens with UDP based transmission, or retrying the transmission of packets, as happens with TCP. The chunk elimination approach is well matched with SVC video, and these techniques and mechanisms are utilized and presented. An evaluation of the performance is provided, plus a comparison of using UDP or TCP, which are the other common approaches for carrying media over IP. Our main contributions are the mapping of SVC video into BPP packets to provide low latency, low loss delivery, which provides better QoE performance than either UDP or TCP, when using those techniques and mechanisms. This approach has proved to be an effective way to enhance the performance of video streaming applications, by obtaining continuous delivery, while maintaining guaranteed quality at the receiver. In this work we have successfully used an H264 SVC encoded video for layered video transmission utilizing BPP, and can demonstrate video delivery with low latency and low loss in limited bandwidth environments.

Keywords Low latency · In-network adaption · Scalable video · QoE

✉ Stuart Clayman
s.clayman@ucl.ac.uk

Müge Sayıt
muge.sayit@ege.edu.tr

[1] Department of Electronic and Electrical Engineering, University College London, London, UK

[2] International Computer Institute, Ege University, Izmir, Turkey

1 Introduction

In recent years, many proposals and emerging network protocols have been proposed, which are defined as the future network architecture and components. Big Packet Protocol (BPP) is one of the network protocols designed by considering the needs and requirements of future networking architectures and applications. With BPP, it is possible to define and implement application specific networking behavior on the network devices at the level of individual packet or flow [20].

BPP was designed with a specific goal of handling the transfer of packets for high-bandwidth, low-latency applications, aiming to overcome a number of issues current networks have with high precision services [20]. This design goal allows applications to define and implement application specific behavior, supported directly by the network, at the level of an individual packet or a flow. This is done by utilizing functionality built into enhanced network nodes [21]. One of the objectives of BPP is to provide a framework which meets the requirements of these high precision services and applications in line with service level guarantees. To support this, the BPP packets have fields providing meta-data for signaling to the routers to act accordingly during the journey of the packet. The BPP payload is partitioned into header and chunks, and some of these chunks can be dropped during end-to-end transmission, depending on the contents of the header and the load of the network. With this process, called *Packet Wash*, whole packets are rarely dropped, instead, parts of the packet payload - the chunks - are dropped, depending on the network conditions and the contents of the header. This provides one of the most important advantages of BPP: namely, it allows the dynamic adaption of packets during their traversal across the network.

Video transmission is commonly done using either UDP, which sends discrete packets but has unreliable properties, or using TCP, which presents bytes streams at the application layer, and has reliable properties. Each approach has it's advantages and disadvantages. With UDP, the network interaction is packet based, and presents loss at the receiver. The application at the receiver has to deal with any packet loss during transmission. If any resends are needed, the application has direct control over these requests. When using TCP, the network interaction is byte-stream based, and presents no loss at the receiver, but there is observable delay and latency. As the application only sees the stream of bytes, it never sees the packets going over the network, and consequently it has no direct control over the resend mechanism, as this is done in the TCP handler of the operating system. With TCP, all the data arrives at the client, but this can be delayed due to re-transmission. The TCP congestion control algorithms can limit real-time video interactions, with elongated buffer durations. Thereby, the receiving application using TCP, is responsible for dealing with the delay, and this is usually done by implementing some buffering techniques. Currently, the majority of video data sent over the Internet is transferred with HTTP, which sits on a TCP transport, or with RTP, which sits on a UDP transport.

BPP packet processing provides advantages to video streaming applications as encoded video consists of video frames, which can be mapped to packets and transmitted over the network. Video receivers can play the video even if some of these frames are lost during transmission. Lost video frames cause different levels of quality degradation at the client, depending on the characteristics of the lost frames. With BPP it is possible to modify packets during transmission, deleting some of the frames or video chunks from the packets, by considering the current network conditions and constraints, which can be beneficial for delivering the highest perceived quality on the client side [5]. This strategy of BPP, to reduce the payload of a packet, and hence its size, by eliminating specific chunks, has a big impact for video. The adaption of the packet size is done by observing the network conditions and taking into account the meta-data that the application set for each chunk. The strategy is to manage the load on the network somewhat, by reducing the consumed bandwidth, yet keeping the flow of video packets arriving at the receiver, so that there is considerably less stalling. This means cutting out specific segments of the transferred video, rather than dropping packets or retrying the transmission of packets [5].

To make BPP effective for video, BPP needs to be coupled with a video encoder and decoder that can do multiple encodings. One way to produce these quality alternatives is to use scalable video coding, where the video file is encoded such that the encoded file contains one base layer and several enhancement layers. An SVC codec creates video sequences with a number of increasing qualities, from the input video [28]. SVC takes advantage of the similarities between the encoded layers of the same frame, as well as between each frame. The mapping of the SVC layers to chunks in BPP packets has proven to be a good match, and is compatible with the packet modification characteristic of BPP, allowing the deletion of some chunks within packets during their journey [5]. If a BPP packet arrives at a congestion point during transmission and chunk deletion is necessary, the enhancement layers can be removed from the packet. When this packet arrives at the client, the client is still able to play the video.

We chose an H264 SVC encoder as we can create the necessary layered videos for use with BPP. This has proven to be an effective way to enhance the performance of video streaming applications, by obtaining continuous low latency delivery, while maintaining guaranteed quality at the receiver [6]. Although more modern codecs such as H266 are available [35], currently there is no publically available SVC implementations.

Providing low latency in video streaming applications has gained importance in recent times, with the goal of low latency live streaming services able to keep the latency under 1 s [22]. Although current and next generation multimedia systems, such as AR/VR applications, require low latency and high reliability communication, there is a tradeoff between low latency and high reliability, and they are often conflicting aspects [31].

The purpose of this work is to provide both low latency and high reliability together, by using the advantages of BPP's in-network packet processing for end-to-end video transmission. We use the advantages of *Packet Wash* for end-to-end video transmission, such that maintaining video quality is done directly by the network elements, thus providing instantaneous adaptation and bandwidth utilization. In our previous work, we show that the use of SVC with a BPP transport is a promising approach [10]. The preliminary test results given in [6] and [11] show that it is possible to reduce the duration of pauses and the number of lost layers when compared to UDP and TCP. We also showed how an Software Defined Network (SDN) controller can facilitate the *Packet Wash* mechanism [11]. In [7], we present the effects of *Packet Wash* on SVC video in limited bandwidth environments. In this work, we enhance our previous work by presenting a discussion and performance results about how different packing strategies for mapping the video layers into packets can affect the QoE on the client side.

This paper presents the techniques and approaches used to map layered SVC video streams, into a set of packets for network transmission when using the BPP. With BPP, the packets have to be constructed in a different way to when using raw UDP, or RTP over UDP, or HTTP over TCP, as the BPP payload is partitioned into a header plus a number of data chunks, rather than being one set of bytes. During the evaluation, we directly compare the performance of BPP, against UDP and TCP, which, as stated, are the commonly used transports for current video transmission.

To-the best of our knowledge, this is the first paper that comprehensively shows the required mechanisms to transfer SVC video data with BPP and the impacts of various packing strategies on QoE. The contributions of this paper are threefold: (i): we present the new processing mechanisms and techniques which are utilized for multi-layer H264 SVC video, (ii): we show how SVC video data is mapped into BPP packets, and introduce the additionally required fields for transferring SVC video data, and (iii): we show how the BPP packing strategies affect QoE under different network conditions.

The experimental results show that the impact of these strategies changes with respect to the changes in the available bandwidth capacity The mapping of SVC video into BPP packets, using different packing strategies, provides a low latency, low loss delivery of the video by creating a suitable packet structure amenable with BPP. Given that the network can eliminate chunks of data during transmission, we present the changes in the data volume of each video layer. We also discuss how the client takes this packet structure, and reconstructs a valid H264 stream. The paper shows the rationale for using BPP for video transmission, with a particular focus on getting the video into packets, and converting the packets back to a video stream. This is followed by a performance evaluation, and our conclusions and further work.

2 Background and Related Work

2.1 BPP Characteristics and Related Works

Nowadays, TCP is the most preferable transport layer protocol used in video streaming systems. Most of these systems use HTTP Adaptive Streaming (HAS) which enables quality adaption while relying on the reliability of TCP and utilizing HTTP web caches. HTTP is a protocol sitting on TCP [1], and common techniques for transmitting video, such as those used by CDNs, include HTTP Adaptive Video Streaming which has become a de-facto streaming technology. To deal with varying network behaviour and changes in loss patterns over time, the MPEG-DASH standard has been devised. Dynamic Adaptive Streaming over HTTP (DASH) [24] is a standard developed by the MPEG research group, to provide interoperability between the elements of various HTTP Adaptive Video Streaming systems. The system is designed to send segments of video, of a few seconds length, at the requested quality. It starts with the lowest quality, and progressively goes to higher qualities if the receiver observes that segments arrive in a timely manner, and concludes that congestion is low and thus network bandwidth is available. The receiver adapts the quality of the video being sent dynamically, by requesting segments of a different quality, as needed. The receiver puts the received segments into the decoder for viewing. To deal with the various bandwidths available, DASH encodes a video multiple times, each with different visual qualities. Higher quality video means a higher data rate, bigger files, and more bandwidth being used.

DASH is a pull-model where the receiving client actually makes requests for a large number of separate files. Consider a video stream of 2 h duration, this equals 120 min, which is 7200 s. Suppose each segment is 2 s long, which for 7200 s of video, produces 3600 files. As stated, each video is encoded into multiple qualities, so DASH will produce files for each encoding, each with a different bitrate, graded from the lowest to the highest. If there are 3 quality alternatives for a 2 h video, a total of 10800 files will be produced, viewed as low, medium, and high qualities. As a side-effect there is a large meta data file which contains a list of all the other files, with their qualities and their starting times. A consequence of this approach is that although DASH is reliable and can use HTTP caches, it has some significant down sides. It produces many little files on the server, and those files are fixed sized time segments. If the end-to-end system need to request segments longer than 2 s, it would require a full re-encoding of the original video into a set of new files of the desired length. Furthermore, DASH been demonstrated as being poor for low latency aspects, and this is a topic of research with DASH [22].

The QUIC protocol, which provides HTTP over UDP instead of over TCP, was introduced to improve the quality of web services [3]. It has often been suggested as another good approach for video transmission, as it transmits over UDP rather than TCP. However, just because QUIC sits over UDP, does not necessarily make it an ideal solution. Rather, it been found to have some drawbacks. The authors of [30] found that there is no evidence for any QoE improvement using QUIC, over normal HTTP, when streaming YouTube videos. In [25], it was observed that QUIC is too reliable for real-time video traffic. Also, QUIC sometimes performed worse than TCP for video streaming.

The emerging network technologies and the increasing demands of future multimedia applications make researchers work on new protocols and systems that utilize new technologies and support new applications. BPP was first introduced in 2018 [20], and the use of the *Packet Wash* process, where chunks in packets are dropped, was shown to minimize latency in [15]. The importance of new high precision services in emerging networks, and how the use of BPP can provide such facilities was presented in [12], and was used for Time Sensitive Networking in [23], and for Mobile Edge Networks in [14]. None of these papers focus on multimedia transmission over BPP, nor presented any results related to that. Although BPP was designed to be an effective mechanism for media delivery, our work was the first actual implementation of the transmission video using BPP, and for observing the effects of using BPP for video. With BPP, packet losses are expected to be minimized due to its unique characteristics that allows network nodes to dynamically adapt the size of the packets during transmission. TCP and UDP have different advantages and disadvantages. While UDP has unreliable communication, TCP provides reliable communication. BPP provides a way to make some parts of the payload to be reliable, and other parts to be removable in the network. This new type of protocol may use the best characteristics of both TCP and UDP. As an alternative protocol to TCP and UDP, BPP can be used in video streaming systems, and it provides various advantages. If UDP is used as the underlying transport layer protocol, other protocols, such as RTP or RTSP that supports media streaming are combined with UDP. RTP is a protocol based on Application Layer Framing [4], and the RTP/RTCP protocols provide a packet format that allows the participating entities to communicate information such as payload type, lost packets, timestamps and sequence numbers [13]. RTSP, which is a stateful protocol used for video delivery [26], is a presentation-layer protocol that lets end-users command media servers via pause and play capabilities. RTSP uses RTP as it's transport protocol [27] to carry the streams. In this context, BPP is a transport protocol, and should not be viewed as an alternative to RTP, which is conceptually the next layer up.

When using BPP for layered video transmission, the clients receive a continuous stream of playable video, having low levels of outage, even with limited bandwidths. In [11] it is observed that managing the network jointly by using SDN and BPP provides higher effective bandwidth utilization, by not having packets that are sent and then dropped, i.e. wasted. It shows that the network node can adapt the packet structure, by eliminating chunks, and thereby adapting the video quality, whenever such a process it is needed. The packet structure for BPP has a header plus some metadata, and a number of chunks. The header holds each chunk size, the chunk offset in the packet, a significance value, and some BPP commands for each chunk. The full details of the structure of a BPP packet, and the definitions of the main blocks, can be found in [20].

2.2 Low Latency in Video Streaming

Providing low latency when delivering video streams is a challenge. One way to provide this is to adapt the video quality during transmission, on the basis of the changes in the network conditions. Doing quality adaption when streaming video can be done in various places. Before client-side adaption became popular, which is the main characteristic of HAS systems, doing adaption on the server side was popular. HAS systems are widely used today and it's adaption mechanism enables users to get the best possible QoE for the given network conditions. The quality adaption is done by the clients by considering the network conditions on their side and internal parameters such as buffer fullness. The success of these systems show that the quality adaption based on the observed network conditions works well for QoE [29] and bitrate adaption [2]. Server-side adaption is generally used by the systems which are based on UDP transmission, combined with an RTP payload.

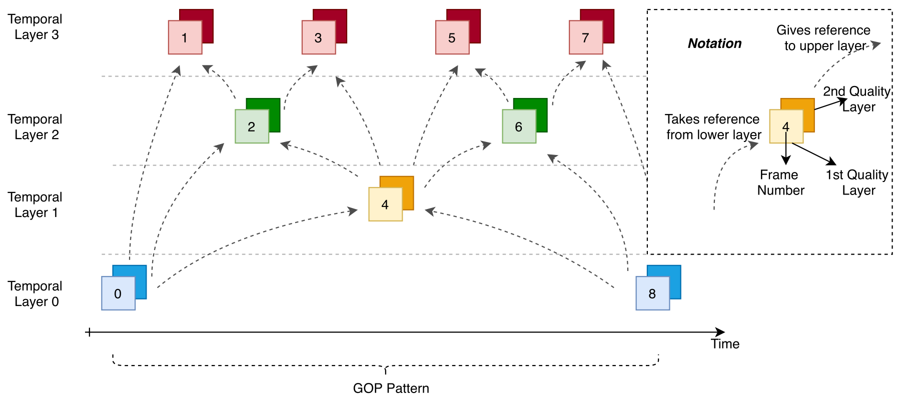


Fig. 1 An example frame structure showing the temporal layers and spatial quality layers and their relationship

Although there are remarkable solutions that adapt the quality successfully, satisfying stringent low latency requirements, especially in live video streaming, is still an issue [18]. The use of HTTP Chunked Transfer Encoding (CTE) with the Common Media Application Format (CMAF) is a promising solution to solve this issue [18, 32, 33]. This approach allows clients to receive smaller parts of the segments and to start playout before the segment is fully downloaded. However, even if low latency is provided, providing high quality and less duration of pauses remains as a challenge [22]. Apple developed LL-HLS for providing low latency service, in which the CTE approach is not used. In [16], the performance results of LL-HLS show that while it provides low latency, the clients send a significant number of HTTP GET requests for those short segments, which may cause an overhead problem [16].

2.3 Scalable Video Coding

To obtain different qualities of the same video file when a non-scalable codec is used, then the video needs to be encoded several times. Hence, the result is one video file for each encoded quality. Using SVC provides an alternative way to obtain different qualities [28]. With SVC, more than one quality alternative can be extracted from an encoded video, all in the same video file. When using SVC, the video is encoded such that the encoded file contains one base layer and a number of enhancement layers. The base layer provides the lowest quality of video, and it does not need any further layers to be decodable. Any enhancement layers are dependent on the layers below them in order to be decodable.

There are different types of quality layers in SVC, called temporal layers, spatial layers, and quality layers. Each frame, regardless of its type, is made up of Group of Blocks (GOB), and these made of the Macro Blocks. Figure 1 shows such a video, with the frame structure of an encoded video and dependencies between frames are illustrated. In the figure, each frame is within a Group of Pictures (GOP), and consists of 8 frames which corresponds to 1 s of video. At the lowest temporal layer, there are only I frames. As seen from the figure, I frames do not use any other frames as a reference. Hence, if we only have I frames the video is decodable. In the figure, there is another enhancement layer type, quality enhancement layers are illustrated. Quality enhancement layers have no effect on frame resolution, it only improves the SNR of the frame. There is another type of enhancement layer called spatial layer, which is not shown for simplicity. Spatial enhancement layers increase the resolution of the video.

The transmission of SVC video has been shown to have some benefits [34], with some researchers having combined the use of DASH with SVC video [17, 19] to use the advantages of the Scalable Video Coding.

In this work we utilize the layered attributes of SVC, and align this with the capabilities of BPP. In previous work, we demonstrated that it is possible to do immediate *in-network quality adaption* of the video, with respect to the changing network conditions, while maintaining a high quality which significantly outperforms UDP in terms of Peak Signal to Noise Ratio (PSNR) and duration of pauses [10], and outperforms TCP with respect to duration of pauses [11], as BPP enables such in-network adaption due to its feature that allows for adjusting the size and the content of the packets during their journey on the network. In [7], we show that different packing strategies have a varying impact on the number and the size of washed chunks. In this work, we enhanced the previous work and show the relation between packing strategies and QoE.

3 Mapping Video Streams to BPP

To make BPP effective for use in carrying video, BPP needs to be coupled with an encoder and decoder that can do multiple encodings for the same frame [5], and utilize an in-network function to process the BPP commands [11]. The video content should be arranged and encoded in a way suitable for BPP. The packet creation method on the sender should be aware of the BPP transmission process, and packets should be constructed so that some of the chunks of video can be removed during transmission. This approach will allow the clients to play the video with an acceptable quality.

Conversely, using BPP in the most simple way, some BPP chunks can be dropped from a packet and the smaller packet be delivered, which is obviously different to having the whole packet being dropped. However, if blocks of video are placed in BPP chunks, without any consideration of the video and the receiver processing, then if some of those chunks are dropped during transmission, the receiving client will see loss of video data. This loss will still have to be dealt with by the receiver, and this has a very similar impact to loss of a packet. Used in this way, BPP provides very little benefit.

We now discuss the details of the structure of an H264 video stream, and present the segments inside the stream, called Network Abstraction Layer (NALs). Then we show how the different layers are selected and how the data for these layers is mapped into BPP packets. This approach ensures that there is always at least one

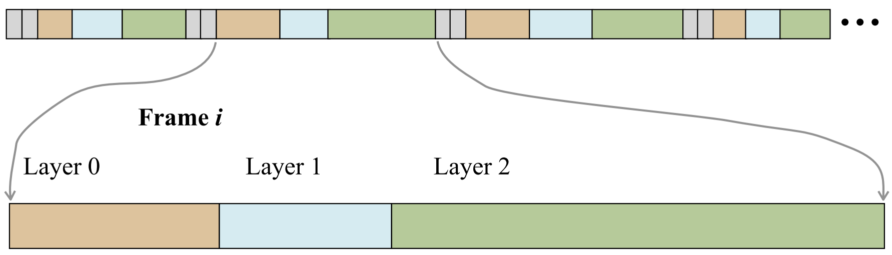


Fig. 2 H264 sequence of NALs. NONVCL (*grey*) VCL (*3 layers*)

chunk that can be delivered, for each packet, even if other chunks are removed. We also show how the client receives the BPP packets and reconstructs the NALs to create a valid H264 playback stream.

3.1 SVC H264 Structure

The structure of an H264 file, that utilizes SVC, which we have used for transmission over BPP, is presented here. An H264 file has a sequence of NALs which hold 2 main kinds of data: *VCL*—Video Coding Layers, which contain the encoded video data; and *NONVCL*—Non Video Coding Layers which hold meta-data related to the current part video stream. 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 frame. The video can be conceptually viewed as 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. In general, we could use an encoding with any number of layers, which would produce more NALs per frame.

In a H264 file, the conceptual layered structure is placed into a byte stream, and presented as a sequence of ordered sub-streams, with the VCL NALs for each frame appearing one after the other. NONVCL NALs with meta-data precede these video data NALs. Figure 2 shows this sequence and how there are NONVCL NALs mixed with the VCL NALs. The 3 NALs for the video data can be seen. Our process reads these 3 NALs from the file and processes them for transmission. This process is described in the next section.

In Fig. 3 the major components of the video frames, the Macro Blocks, are encoded with a layered coding as illustrated. In the figure, we see that there are 3 layers: Layer 0 (the base layer) with the lowest bitrate data, Layer 1 with medium bitrate data, and Layer 2 with the highest bitrate data. These layered blocks will become the parts for the BPP packets.

3.2 NALs to Packets

Within the video stream, shown in Fig. 2, there are a number of NONVCL NALs which are interspersed between the VCL NALs. These are generally small in size

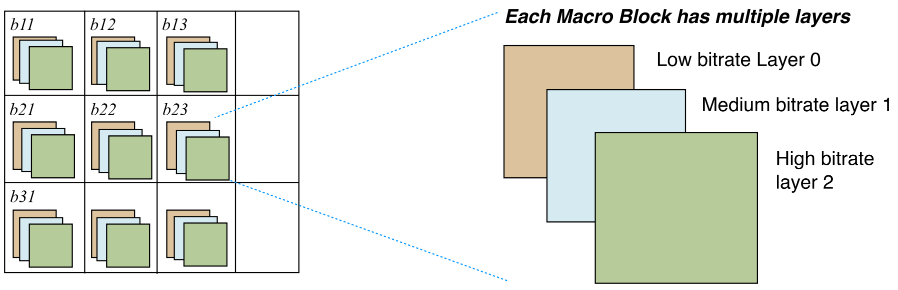


Fig. 3 Conceptual frame: each macro block has multiple layers

and can easily be placed into a packet. In our approach, we send the NONVCL NALs as a single BPP chunk. As we have a 3 layer video, we see 3 VCL NALs for each video frame in the video stream. For nearly all videos, the frame size and the NAL size is much bigger than a standard 1500 byte packet, therefore the sequence of NALs has to be collected for each frame, and then mapped into BPP packets. This mapping task is done using 3 main processing phases for VCLs:

1. It collects and enumerates the sequence of NALs from the video stream. In this phase an H264 stream processor collects 3 VCL NALs from the input;
2. It splits the collected 3 NALs for each frame, and packs them into a array of intermediate objects, which we call ChunkInfo objects; and
3. It takes the array of ChunkInfo objects and converts them into the BPP packets, ready for transmission.

Step one of the processing requires an H264 stream reader and processor which can return the bits and bytes of the input data, and return each NAL.

Having an intermediate ChunkInfo form provides a degree of flexibility and adaptability while processing the NALs, before they get mapped into packets and sent to the network. For each ChunkInfo object which may contain 1 or more whole NALs (usually NONVCLs as they are small), or 1 NAL that is fragmented (usually VCL), we keep the following data:

– the NAL type: NONVCL or VCL
– the NAL number within the stream
– a count of the NALs held
– a fragment number
– is it the last fragment
– chunk size (in bytes)
– the chunk payload bytes
– no of bytes remaining in a packet

This ChunkInfo object allows us to do analysis on the volume of data than can be placed into a packet and allows for timing calculations. It also facilitates a number of packing and multiplexing strategies for the packets, as there are a number of ways in which the VCL NALs can be put into BPP packets.

3.2.1 Packing Strategies

Step two of the process utilizes a number of packet filling strategies we have devised to collect video data from the VCL NALs and to pack these 3 sub-streams of video data into the ChunkInfo objects, and then into the chunks of a number of BPP packets. There are currently 4 strategies that we use. Each strategy has different characteristics, which gives a tradeoff between different QoE parameters:

- *Even Split*—which splits the incoming data evenly into equal size chunks. When we have 3 NALs we allocate the same number of bytes to each chunk. There is at least one removable chunk in all of the packets with this approach, being mostly small in size. This allows the shrinking of packets, while still having useful chunks in the delivered packet. This strategy provides fine-tuning capabilities in quality adaption.
- *Dynamic Split*—which 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. Similar to the Even Split strategy, there is at least one removable chunk in all of the packets. The difference with this approach, is that the number of the packets constructed is fewer. This characteristic allows packet delivery to the clients in a shorter time, causing smaller duration of pauses when the bandwidth is enough.
- *In Order*—which creates chunks in the order they come from the NALs. Data from just one NAL will fill the packet, until the last chunk, which may be smaller. All the other chunks are empty. In this strategy, if the network node removes a chunk, then the packet will be delivered to the client with no video content chunks. The can cause delays waiting for video data.
- *Fully Packed*—which is similar to In Order, except that on the last chunk of a NAL, the remaining data is allocated to the next NAL. The flexibility in quality adoption of this strategy is between the In Order approach and the other strategies. The packets might have a big sized chunk or a small sized chunk that is removable, or might even have no chunks that are removable.

Different characteristics of the strategies presented above, cause different performance in terms of QoE parameters. With In Order and Fully Packed strategies, the base layer video is put into the first set of packets. Hence, if the bandwidth is limited and the packets are delayed, the clients can play the video as soon as they get the packets with the base layer, which provides small duration of pauses. On the other hand, because the layers are distributed into the packets with Even Split and Dynamic approaches, in-network quality adoption can be fine-tuned and done better.

Figure 4 shows the structural mapping of the video NALs into the chunks for each packet for the *Even Split* and *Dynamic Split* strategies, Fig. 5 shows the mapping for the *In Order* strategy, and Fig. 6 shows the mapping for the *Fully Packed*

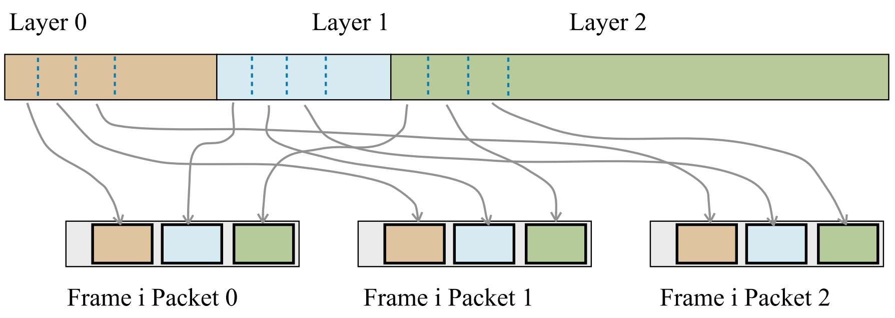


Fig. 4 Even split and dynamic split for 3 NAL layers

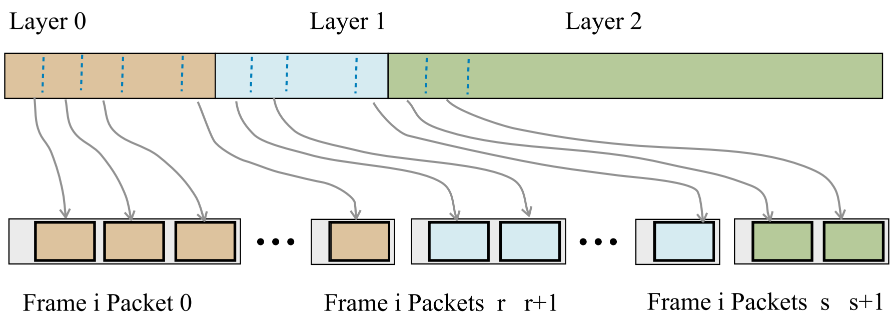


Fig. 5 In order for 3 NAL layers

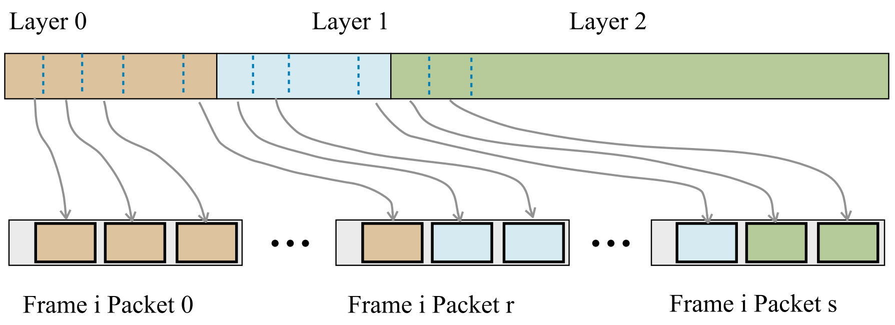
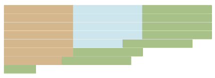


Fig. 6 Fully packed for 3 NAL layers

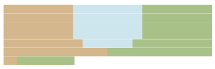
strategy. In these schemes, if the NAL is bigger than the chunk space allocated by the strategy, then that many bytes are taken from the NAL and put into the chunk, and the remaining NAL data is kept for the next chunk. If the remaining NAL data is smaller than the chunk space, then the chunk will only contains those bytes. For NALs which are split this way, we label them with a fragment number. This allows reliable reconstruction at the client. If one of the NALs has no more data, as we have allocated all of its data to chunks, then we still allocate a chunk, together with its meta-data, but we set its chunk size to 0.

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



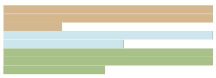
(a) Even Split Strategy

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



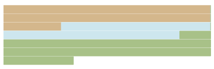
(b) Dynamic Split Strategy

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

Fig. 7 Different packing strategies have different chunks. 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

Consider the effects of these 4 different packing strategies on the sizes of the chunks and the structure of the BPP packets. The data presented in Fig. 7 shows how the different strategies produce different packet structures and sizes. We use a 1500 byte packet, providing a content size of 1472 bytes. This allows for the BPP header and the relevant BPP meta-data, before allocating data to the chunks of video. In this example, the original sizes for 3 VCL NALs are as follows: *Layer 0*: 3232 bytes, *Layer 1*: 2232 bytes, and *Layer 2*: 3527 bytes. Figure 7 has a table and a graphic for each packing strategy. It shows the sizes of the chunks for each packet in the table, and a visual showing the proportional size and relevant colour for the parts of the different layers.

3.2.2 Setting Significance Values

During the ChunkInfo creation process, the *significance value* of each chunk is set. The significance values in the BPP packets shows the importance of each chunk within the packet. This value provides the network node with the information needed to either keep the chunk, or wash it away. As we use BPP to trim packets during transmission using the *Packet Wash* process, the chunk removal policy is done by considering both the chunk's importance to the video content and the available bandwidth.

In this paper, we set the priorities of the chunks by taking into account the chunk impact on both the decoding process and on QoE. The VCL NALs that have the highest importance to the video, are: (i) those that hold the base layer, (ii) all layers of an I frame, and (iii) all NONVCL NALs. These are set with the lowest significance value, which indicates such importance to the network node. This means that those chunks should not removed during transmission. The values selected for our experiments, are shown in Table 1, in Sect. 5.

3.3 Packets to NALs

Here we present how the client takes packets from the network with the BPP format, analyses the structure to determine if any chunks have been eliminated during transmission, and rebuilds the NALs for the video stream. The task is the reverse of the 3 phases of the sender. The first step is to convert the packets back into the intermediate ChunkInfo objects. Second, the chunks of video are extracted, demultiplexed, and turned back into a sequence of NALs. Third, is to create a valid H264 SVC stream.

As described in Sects. 3.2.2 and 4, a network node can trim out chunks from a BPP packet if the network conditions are such that there is a bandwidth limitation / bottleneck and the node determines that chunk trimming is necessary. The client has to accommodate and handle any of these missing chunks.

3.3.1 Handling Missing Chunks

To deal with the situation where chunks of video are removed by network nodes, a process has been designed and built. If a chunk is missing, it means that the NAL which contains that chunk cannot be rebuilt in a valid way, and so even the other received chunks from 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. However, there are also dependencies between the layers and the consecutive frames. 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 at those layers, particularly if they never made it from the I frame. If necessary some extra NALs which have lost their dependent NAL are *cleaned* away also. The resulting video provides a valid stream for the decoder, and a foundation for the experiments presented next.

From an SVC video, it is possible to extract the different layers as needed, by reading the relevant NALs from the H264 stream [28]. Our results herein show that SVC video is highly compatible with the packet modification characteristic of BPP, as it allows the deletion of some chunks from selected packets during transmission. If a BPP packet arrives at a network node during its journey, where the network conditions are such that there is a bottleneck and network node determines that chunk deletion is necessary, then in this situation some chunks containing enhancement layer data can be removed from the packet. When a packet with deleted chunks arrives at a client, a carefully designed process of packet demultiplexing, using meta-data specification and NAL reconstruction, allows the client to still be able to play the video.

4 Using BPP for Video Transmission

Following the discussion on how the NALs of the video stream are mapped into BPP packets, in this section, we present the BPP packet structure and give and explanations about its fields. As stated before, just by using BPP, does not in itself

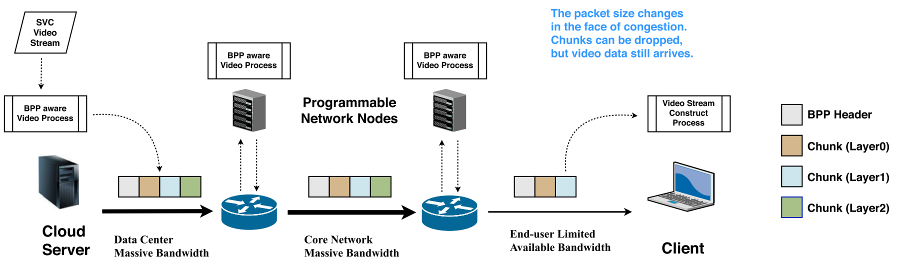

SVC Video Stream
BPP aware Video Process
Cloud Server
Data Center Massive Bandwidth
BPP aware Video Process
Programmable Network Nodes
BPP aware Video Process
Core Network Massive Bandwidth
The packet size changes in the face of congestion. Chunks can be dropped, but video data still arrives.
End-user Limited Available Bandwidth
Video Stream Construct Process
Client
BPP Header
Chunk (Layer0)
Chunk (Layer1)
Chunk (Layer2)

Fig. 8 BPP packet processing path

guarantee improved behaviour or performance. Although the BPP structure enables flexibility of packet size change through trimming during transmission, the use of these fields for streaming video needs to be be defined, as well as any enhancements that should be done. We now present a more detailed view of using BPP for video transmission, and show the structure of the BPP packets, including the header and metadata blocks, and how it can be used for carrying multi-layer video.

4.1 BPP Packet Path

In order to execute the BPP packet processing, BPP enabled switches are required. BPP enabled switches can also be facilitated by using a virtualized network function with this characteristic or by using OpenFlow enabled switches and an SDN controller. In our previous work, the SDN controller decided the updates / modification of the BPP packets and the switches then forward the BPP packets processed by the controller. In this work, a virtual network function which is part of a virtual router does this evaluation. In Fig. 8, the general outline of this scheme is illustrated.

When BPP is used for the transmission of video, it has a large impact on the applications that send and receive video. The server has a BPP aware process which does the mapping of the video stream to packets, described in Sect. 3. The BPP aware network nodes execute the *Packet Wash* process, and may reduce the packet size, by trimming, if there is congestion. Finally, the client uses the BPP aware video reconstruction process to build a valid video stream.

4.2 BPP Packet Structure

BPP extends the IP packet structure by adding some additional fields which are collectively known as a BPP block. The full BPP packet structure is given [21], but a brief explanation of the fields of a BPP block follows.

4.2.1 BPP Block

A BPP block includes user or application-related information to provide guidance to the routers for packet processing. A BPP Block contains a *Command and Metadata Block*. While the *Command* part carries commands and their conditions and

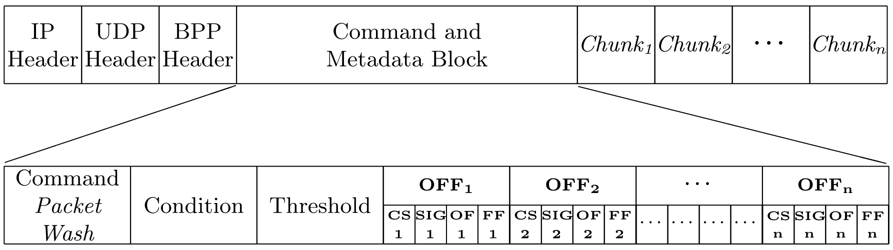


Fig. 9 BPP structure in an IP packet

parameters, the *Metadata* part holds additional metadata. These parts are optional, as the applications may need to use only one and not the other, as described in [20].

4.2.2 BPP Header

The BPP Block Header describes the overall structure of the BPP Block. The BPP version and block length, indicating the length of the BPP Block, are given as fields within the header.

4.2.3 Command and Metadata Block

The Command and Metadata Block carries information related to command–condition pairs, the actions, as well as additional metadata for the current packet. The structure of the Command and Metadata Block is shown in Fig. 9, which presents the fields. These are described in more detail soon.

For our work we utilise the *Packet Wash* command, but the design is more general [20]. An example of another condition-command pair includes: “If the delay exceeds a certain threshold, dropping the packet payload”. Further information about packets or the flow, such as security and accounting related information, can also be carried within the Metadata fields.

4.2.4 Packet Wash Command

The *Packet Wash* command was proposed in [21], as an extension to BPP, to partition the payload into smaller chunks and enable the removal of some chunks during transmission. This approach is designed to increase bandwidth utilization due to its mechanism that shrinks the packet, instead of dropping the whole packet. It operates by signaling the node with information about the *significance* of (or the relationship between) the chunks, and then dropping lower-priority chunks from the payload according to the information carried in the packet header. These lost chunks cannot be recovered, but it does allow some information to be usable at the receiver [21]. It is this process which we utilise in our work.

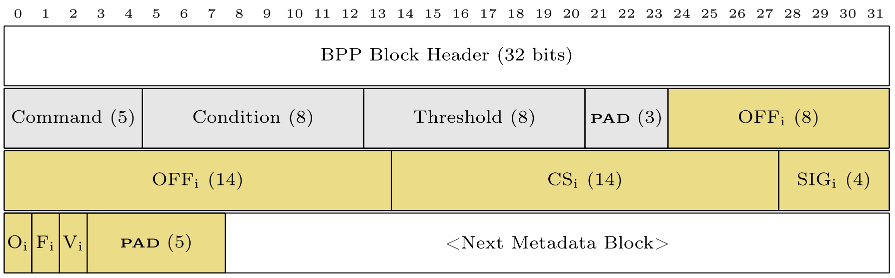


Fig. 10 Bits and bytes layout

4.3 BPP Fields for Video Streaming

In order to stream layered video with BPP, we have repurposed some of the fields of the *Command and Metadata Block*, to hold the video related data in the chunks carried within the packets. We have extended the standard BPP fields with new fields to carry video specific data. The fields for transferring video over BPP are given in Fig. 10, showing the bits and bytes of each field in the BPP packet. The implementation has the following layout:

– The *BPP Block Header* is 4 bytes (32 bits).
– The *Command Block*, shown in grey, is 3 bytes (24 bits) including: 5 bits (Command) + 8 bits (Condition) + 8 bits (Threshold) + 3 bits (padding).
– The *Metadata Block*, shown in yellow, is 6 bytes (48 bits) and has one entry for each of the chunks in a packet. It includes: 22 bits (OFF_i) + 14 bits (CS_i) + 4 bits (SIG_i) + 1 bit (OF_i) + 1 bit (FF_i) + 1 bit (V_i: VCL or NONVCL) + 5 bits (padding)

In the *Command Block*, the **Command** specified is *Packet Wash*. There are other command types available.

The **Condition** field is used to pass information as to whether any chunk could be removed from a packet. To signal the network node to decide the conditions under which the chunks should be removed from a packet, the server can set a *Condition* value. The server can calculate a heuristic number, based on the encoding characteristics of the video, and this is used as the *Condition* value.

The **Threshold** value represents the significance value below which the chunks cannot be further dropped. This value is determined considering the importance of the video frames and the layers. Each layer and frame type pair is given a value from 1 to 15, as seen in Table 1, with the default Threshold value being set to 5.

The *Offset* block contains **OFF_i (22 bits)**, which was introduced in [20], and is utilised here for supporting video streaming applications. The details of the *Offset* structure is shown in Fig. 11, and presents the fields used for each chunk. The Offset field is divided into three subfields: *NAL count*, *NAL number*, and *FragmentationNo*. The field *NAL count* (5 bits) shows how many NALs are within a packet, *NAL number* (12 bits) shows the current NAL number carried in a chunk,

$\mathbf{OFF_i}$ (22 bits)		
NAL Count (5 bits)	*NAL No* (12 bits)	*FragmentationNo* (5 bit)

$\mathbf{CS_i}$ (14 bits)	$\mathbf{SIG_i}$ (4 bits)	$\mathbf{OF_i}$ (1 bit)	$\mathbf{FF_i}$ (1 bit)	$\mathbf{V_i}$ (1 bit)

Fig. 11 Offset structure

for complete or fragmented frames, and *FragmentationNo* (5 bits) indicates the fragmentation number that chunk represents, in case the frame is fragmented. As some frames might be fragmented into the more than one packet, due to their size, the Fragmentation number is used to combine fragments on the client side. As the NAL number is only 12 bits, the largest value it can hold is 4095, so the client is responsible for tracking if the value wraps. Similarly, the FragmentationNo is only 5 bits, with the largest value being 31. Again, the client has to track if wrapping occurs.

The *Offset* block also uses the following fields to keep information about layered video: **CS**$_i$ **(14 bits)**: holds the size of the current chunk; **SIG**$_i$ **(4 bits)**: holds the significance value of the chunk; **OF**$_i$ **(1 bit)**: is set to 1 if the chunk is dropped; **FF**$_i$ **(1 bit)**: shows if the current fragment is the last one; and **V**$_i$ **(1 bit)**: shows if the current NAL is VCL (video) or NONVCL.

The server is responsible for creating packets using the described packet layout, the network node needs to inspect the relevant fields during processing in order to execute the specified command, and finally the client needs to deconstruct the packet to extract the video.

4.4 Packet Processing

During the transmission of the packets, chunks should be dropped if the total amount of data transferred in a specific time period exceeds the available bandwidth. When the node receives a BPP packet, the process decides whether it is suitable to send the packet onwards without modification by checking the size of the packet. The BPP aware video process calculates the number of bytes transferred within 1 s. If this calculated value exceeds the available bandwidth, then the network node starts the packet processing and evaluates against the current available bandwidth on the links.

The process checks each packet to determine which chunks can be removed from the packet. The rate at which chunks should be reduced is decided according to the available bandwidth measurements. Depending on this, the upper limit for the packet size is calculated. Chunks are then deleted by considering the *significance values* of the chunks, so that the packet size gets under the specific limit. Chunks whose *significance values* are lower than the threshold cannot deleted in any situation. The process proceeds to remove chunks until the packet size is reduced. The trimmed packet is forwarded onwards.

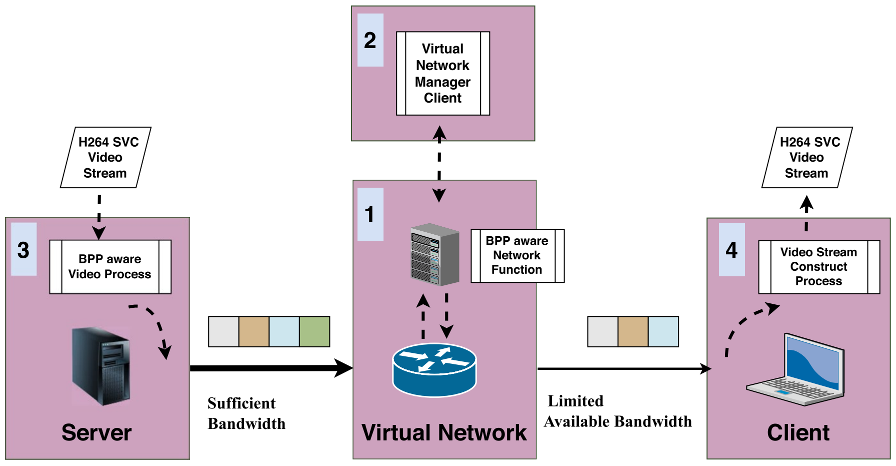


Fig. 12 System setup

Using the packet structure and the processing techniques described, allows the BPP aware video process to directly provide *in-network adaption* of video streams. The evaluation of this is presented in the next section.

5 Performance Evaluations

In this section we present an evaluation of sending an SVC video, using BPP, over links with different bandwidths, using the 4 different packing strategies that were described in Sect. 3. We present results showing comparative data with respect to using a UDP transport, which is utilized by RTP, and to using a TCP transport, which is utilized by HTTP, together with the observed QoE parameters, comparing those 4 packing strategies.

5.1 Evaluation Setup

5.1.1 Processing System

In order to execute the BPP packet processing, BPP enabled switches are required. In this study, BPP enabled switches are facilitated by a BPP-aware Virtual Network Function (VNF) built using the Very Lightweight Software-Driven Network and Services Platform (VLSP) platform [8, 9]. The network function receives the BPP packets and decides which updates/modification are needed, and the switches then forward the packets processed by the VNF. The general outline of this scheme is shown in Fig. 12.

VLSP is an open-source platform for creating and evaluating experimental deployments of heterogeneous softwarized network resources, with flexible and service-aware management facilities. Such unified environments as VLSP, have

Table 1 Significance values used for the experimental video

Layer	Frame type	T value	SIG value
L0	I	T0	1
L1	I	T0	2
L2	I	T0	3
L0	P	T1	2
L1	P	T1	7
L2	P	T1	12
L0	P	T2	3
L1	P	T2	8
L2	P	T2	13
L0	P	T3	4
L1	P	T3	9
L2	P	T3	14
L0	P	T4	5
L1	P	T4	10
L2	P	T4	15

efficient and distributed management facilities that demonstrate scalability, reliability, and adaptability to the dynamic conditions in terms of resource availability and changing service and infrastructure requirements. To assist with the deployment of VLSP systems, we developed a distributed facility for testing, evaluating, and experimenting with the management of such environments. VLSP exhibits the following properties: (i) it provides a complete integrated management platform for Software Defined Infrastructure environments as a basis for experimentation; and (ii) it is distributed and scalable, across multiple hosts, adopting lightweight virtual entities, making it suitable for testing a wide-range of network functions and management features over diverse topologies.

In the system deployed here, in Fig. 12, the following sub-systems are utilized:

1. one executing VLSP with a BPP-aware network function implementing the packet processing functionality described in the previous section;
2. one with a Virtual Network Manager Client which acts as a VLSP controller, sending commands to the VLSP manager;
3. a BPP sender, which implements the SVC video reader and builds BPP packets in the specified format;
4. a BPP receiver, which receives BPP packets, and reconstructs a valid video stream while handling any missing chunks.

In order to evaluate the system performance, we stream video between a server and a client and measure several QoE metrics with different bandwidth settings. For this purpose we have selected the commonly used *Foreman* video,[1] which has been encoded with one base layer and 2 enhancement layers. The bitrate of the base layer is 204 Kbps, the bitrate of base and the first enhancement layer is 488 Kbps, and the bitrate of the base and two enhancement layer is 1094 Kbps. The VLSP network function implements the BPP *Packet Wash* function, and trims packets if the bandwidth is limited. In each experiment, the *available bandwidth* is passed to the network function as an argument.

In Table 1, information about the video layers and frame types used in these experiments is given. In the test scenarios, when considering the importance of the **frame type** and the **layer**, we set a *significance value* for each of them. A significance value **SIG**, given in columns 4, 8 and 12 of the table, is placed by the server into the SIG_i field of a packet, shown in Fig. 11, for the *i*th chunk. As the I frames (with SIG 1, 2, and 3) and the base layer (L0) (with SIG 1–5) must always be received by the clients, in order to be able to play the video with continuity, the threshold value **Threshold** is set to 5. This allows the network function to trim chunks with a significance value above 5, if there is not enough bandwidth.

5.2 Comparative Experiments with UDP and TCP

In order to observe the advantages provided by BPP, we conducted several experiments with BPP, UDP and TCP in our previous work [6, 11]. In those experiments, UDP and TCP packets are filled with the maximum number of bytes that the packet can carry. The layers are put into the packets sequentially, with the base layer being put first. In order to provide a fair comparison, the *Fully Packed* packing strategy is used in BPP transmission, which is the most similar packing strategy to that used in UDP and TCP experiments. For the tests, the bandwidth was set to 0.5 Mbps, 0.8 Mbps, and 1.5 Mbps, respectively. In addition to these fixed bandwidth values, 2 other tests were implemented, by having varying bandwidth conditions, where the bandwidth available continuously increases (i.e ascending) or decreases (i.e. descending) over time.

The average received PSNR values were observed for the BPP, UDP and TCP tests. The PSNR of the original encoded video file is 44 dB, and for these tests the values are listed in Table 2. When the bandwidth is 1.5 Mbps, all the clients play the video with the highest possible quality, since with all the protocols no data loss occurs. When the bandwidth is highly limited, with the bandwidth equal to 0.5 Mbps, UDP cannot provide consistent and good quality in all scenarios, and the PSNR value decreases down to 18 dB with the ascending bandwidth experiment. However, the clients using BPP can play the video with a PSNR value of 40 dB, which highlights that the quality of the received video is high.

[1] Xiph.org Video Test Media collection https://media.xiph.org/video/derf/.

Table 2 Average PSNR values (in dB) *Larger is better*

Bandwidth	BPP	UDP	TCP
0.5 Mbps	40	29	44
0.8 Mbps	40	36	44
1.5 Mbps	44	44	44
Ascending	39	18	44
Descending	40	37	44

Table 3 Video playout on the client side (in s) *10 s is full video*

Bandwidth	BPP	UDP	TCP
0.5 Mbps	10	2.4	10
0.8 Mbps	10	4.2	10
1.5 Mbps	10	10	10
Ascending	10	1.7	10
Descending	10	6.7	10

The video playout duration at the client side for the BPP, UDP, and TCP tests for each scenario, are given in Table 3. The BPP and TCP clients received 10 s of video, and these clients could play the whole video for all scenarios. A significant issue is that for most scenarios when UDP is used, the clients cannot play the video during the streaming session. Although, the original video duration is 10 s, the UDP client could only play the whole video for the scenario where the bandwidth is higher than the video bitrate, namely 1.5 Mbps. We observed that the UDP client for the ascending bandwidth received many packets, especially near to the end of the streaming session. However, in this scenario, the video playout duration is just 1.7 s. The reason for this is that even though the client gets many layers, since the packets carrying *I* frames are lost, the received layers cannot be played, due to missing dependencies. Hence, we observe another drawback of using UDP, namely: even when the network transfers the packets, since those packet are not usable, the effective network utilization is low.

The duration of outage metric highlights another performance indicator for network video. The observed durations for all the tests are presented in Table 4. All clients started to play video after 600 ms to minimize the initial waiting time. During the streaming, BPP showed a significantly higher performance than other protocols. BPP has very low outage, for all scenarios, and even for the most limited bandwidth values, the video outage for BPP is extremely good—just 0.2 s. However, UDP has high outage, with the outage being 5470 ms (more than 5 sec) for the 0.5 Mbps bandwidth. This is half the total duration of the video. Also, the TCP clients observed up to 14 s of outage duration, as can be seen from Table 4, which is longer than the duration of the video. Even when the bandwidth capacity is higher than the bitrate of the highest enhancement layer, TCP clients observed outage.

In order to see the performance of TCP combined with quality adaption, in the style of DASH, we did two extra tests. We streamed the video over TCP with just

Table 4 Duration of outages (in ms) *Smaller is better*

Bandwidth	BPP	UDP	TCP
0.5 Mbps	173	5470	14, 396
0.8 Mbps	205	2634	5697
1.5 Mbps	0	0	584
Ascending	0	916	2554
Descending	0	920	4691

a base layer, where bandwidth equals to 0.3 Mbps, and another test with a base and one enhancement layer, where bandwidth equals to 0.5 Mbps. The latency for these tests were 0 and 734 ms. Even with these highly idealized scenarios, where the server sends the video layers compatible with the bandwidth, we see that the TCP client observed latency.

5.3 Observed QoE Parameters with Different Packing Strategies

In this set of experiments, the server constructs packets by using the 4 different packing strategies, from Sect. 3.2.1. The NALs are mapped to packets, and the packets that are constructed for each strategy carry the 3 layers of a frame, as shown in Fig. 7. As seen from the graphics and the tables in this figure, the number of packets and the size of each chunk differs for each of the strategies.

As described, some chunks of the layers sent by the server may not arrive at the client side to decode, as they can be *washed* by the network node during transmission. Some other layer data arrives at the client, but these layers are not reconstructed, as missing chunks for those layers mean that the layers are not complete and are invalid. Such layers are *cleaned* when doing to the layer dependency analysis. Successfully reconstructed layers are passed on, and the valid received frames sent to the decoder.

In these experiments, the available bandwidth is set to to 0.5 Mbps, 0.8 Mbps, and 1.2 Mbps, respectively. We set the initial buffering time to 1 s where the available bandwidth is 0.5 Mbps, and to 500 ms, where the available bandwidth values are 0.8 and 1.2 Mbps. In Fig. 13, the graph shows the bytes transmitted for each layer by the server in all experiments. In the following figures, Figs. 14, 15, and 16, the bytes received by the clients for each of the packing strategies are given for the 3 different bandwidth values. When the bandwidth value equals to 1.2, in Fig. 14, we see that the bytes received by each strategy is very similar to the bytes sent by the sender due to there being sufficient underlying network capacity. The received number of bytes changes dramatically especially for the highest enhancement layers for the bandwidth values 0.8 and 0.5 as seen from the graphs given in Figs. 15 and 16. This highlights the *Packet Wash* process, combined with the *significance value*, in action. We see that all the base layer (L0) bytes and most of the first enhancement layer (L1) bytes are received by the clients for each packing strategy is different. The bytes that are received in the first

Table 5 QoE results for 0.5 Mbps

	PSNR (dB)	Duration of outages (ms)	Quality changes
Even split	38.54	927	28
Dynamic	37.13	33	50
In order	37	375	64
Fully packed	36.6	0	52

and last seconds of the experiments might differ because the distribution of the mapping NALs to BPP packets used by each strategy.

Analysis of these results and the graphs highlights that BPP was successful in delivering video streams, and no re-transmission mechanism is needed in these experiments, since all the base layer (L0) packets are received by the clients without any dropped packets. We observe low loss and low latency.

Various QoE values can be calculated on the client side which show more behavioural detail. Although there are small differences observable in the number of received bytes from the highest enhancement layer given in the graphs (Figs. 14, 15, and 16), these only present a partial picture. We have calculated QoE values for the same experimental runs, and these are listed for each packing strategy, in Tables 5, 6, and 7 for different bandwidth values. They show the Peak Signal to Noise Ratio values (PSNR); the duration of outages (the total amount of time the client has no video data to play); and the number of quality changes (the number of times the highest quality seen in a frame changes, e.g L1 $\rightarrow$ L2 or L2 $\rightarrow$ L1 is one quality change).

If the bandwidth is highly limited, i.e. when it equals 0.5 Mbps, all the packing strategies have similar PSNR values. However, the differences in PSNR becomes observable as the bandwidth values increase. As mentioned in Sect. 3.2.1, if the bandwidth is limited, the Fully Packed and In Order strategies may provide smaller duration of pauses, since layer 0 packets are sent first. As seen from Table 4, the clients did not observe any video stalls with the Fully Packet strategy not only with limited bandwidth conditions, but also with better conditions. However, the client experienced severe video stalls with the In Order strategy. These results show that even if the most important layers are transferred to the clients at first, the flexibility of in-network adoption plays an important role for the duration of pauses.

If the bandwidth is slightly higher than the highest bitrate, i.e., when it equals 1.2 Mbps, Even Split strategy outperforms the other strategies in terms of both PSNR and the number of quality changes, with a penalty of limited duration of outages. With this bandwidth, the network node rarely needs to remove chunks, removing only a small part of the packet, if this is necessary. The best packing strategy that suits this condition is Even Split because the smallest layer 2 chunks are carried within the packets that are constructed using this strategy.

The minimum number of quality changes is always obtained with Even Split strategy in all experiments. This observation shows that if there is at least one chunk that is removable in the packets, then the BPP-enabled network node might remove

Table 6 QoE results for 1.2 Mbps

	PSNR (dB)	Duration of outages (ms)	Quality changes
Even split	36.6	878	45
Dynamic	36.6	1169	48
In order	36.4	0	78
Fully packed	36.57	0	48

Table 7 QoE results for 0.8 Mbps

	PSNR (dB)	Duration of outages (ms)	Quality changes
Even split	36.6	147	43
Dynamic	36.6	0	45
In order	36.44	2003	78
Fully packed	36.6	0	43

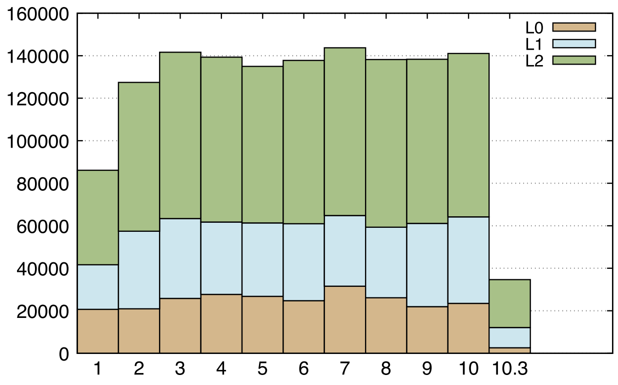
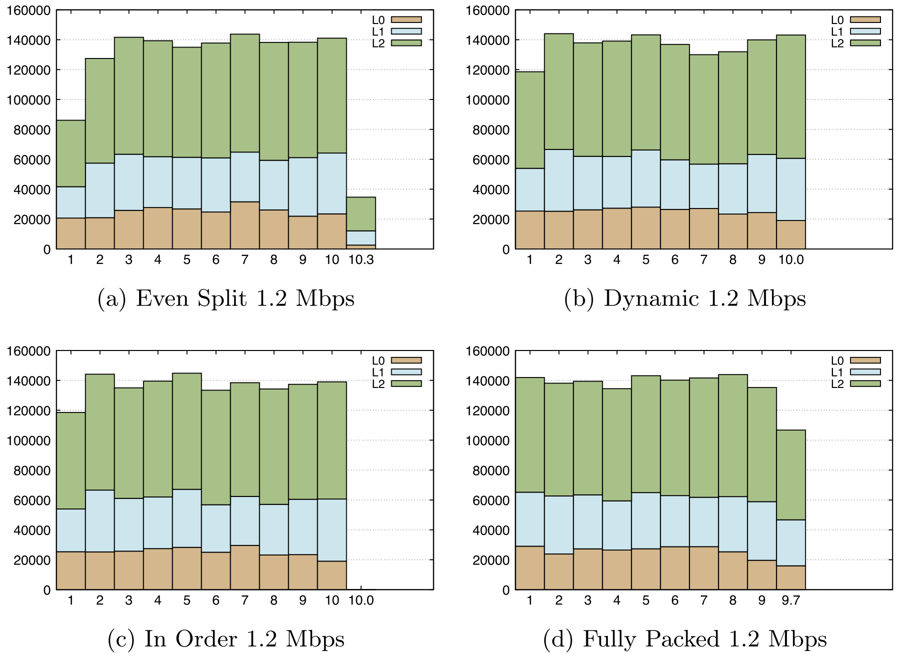


Fig. 13 Transmitted video data—per layer. Time (s) vs volume (bytes)

the chunks belonging the same layer in consistent manner. Dynamic strategy also has lower number of quality changes compared to other strategies since it is another strategy whose packets have removable chunks.

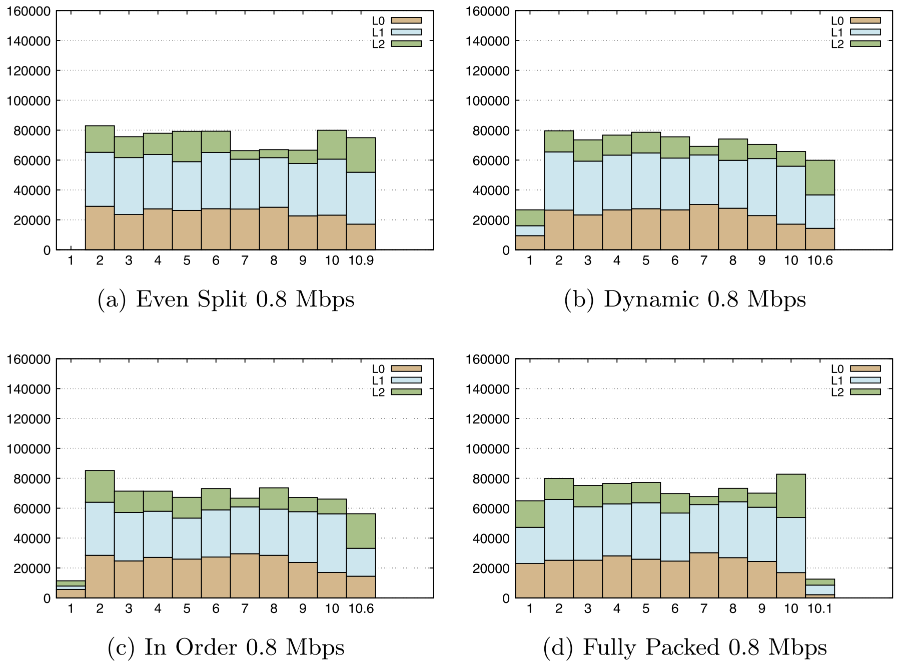
These experiments have shown that BPP provides both reliable and low latency communication while sending scalable video, because it enables in-network quality adaption with respect to the available bandwidth. By having the bandwidth utilization managed carefully, we also observe low loss of whole packets. The quality results for the received video are also demonstrably high. We call BPP a *partially reliable protocol* as it provides a way to make some parts of the payload reliable, and others to be removed in the network.



(a) Even Split 1.2 Mbps (b) Dynamic 1.2 Mbps

(c) In Order 1.2 Mbps (d) Fully Packed 1.2 Mbps

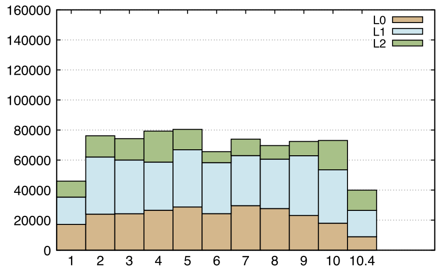
Fig. 14 Received video data. Bandwidth: 1.2 Mb time (s) vs volume (bytes)



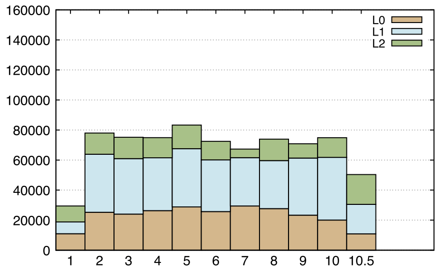
(a) Even Split 0.8 Mbps (b) Dynamic 0.8 Mbps

(c) In Order 0.8 Mbps (d) Fully Packed 0.8 Mbps

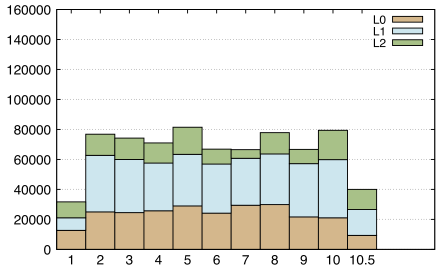
Fig. 15 Received video data. Bandwidth: 0.8 Mbps. Time (s) vs volume (bytes)



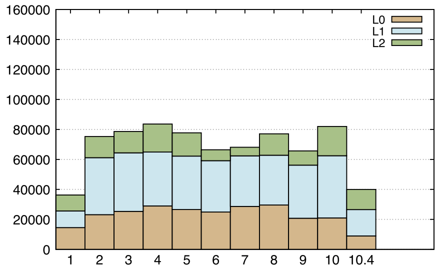
(a) Even Split 0.5 Mbps



(b) Dynamic 0.5 Mbps



(c) In Order 0.5 Mbps



(d) Fully Packed 0.5 Mbps

Fig. 16 Received video data. Bandwidth: 0.5 Mbps. Time (s) vs volume (bytes)

6 Conclusions

In this paper, we have presented an approach for sending H264 SVC video using BPP which has proven to be an effective way to enhance the performance of video streaming applications, while obtaining continuous delivery, maintaining guaranteed quality, and providing low latency at the receiver. The unique feature of the *Packet Wash* feature of BPP is that it enables the adaption of the packet size during transmission, and allows the implementation of in-network quality adaption for video streaming systems.

We have successfully implemented a working layered video transmission mechanism utilizing the BPP packet structure. We have achieved video delivery with low latency while using the BPP Packet Wash mechanism, which we built to remove chunks from the packets as they passed through the network in order to reduce the volume of traffic. This is unlike UDP or TCP, which has to drop whole packets in the face of limited bandwidth. The transmission of this SVC video requires application specific information to be carried in the BPP packets so that the decision as to which chunks should be deleted, and in which order, can be made by considering the bandwidth and the received quality. It shows the advantages provided by a new transport layer protocol which has new features.

It became clear from our work that using a protocol like BPP and layered video streams are complementary. BPP, which supports *Packet Wash*, can cut chunks from the layered video packets in a far more refined and adaptable way than just dropping whole packets as UDP and TCP do, thus it improves the QoE at the receiver.

Furthermore, the use of SDN or Network Function Virtualization (NFV), rather than having special router upgrades, gave a level of flexibility to the solution which is necessary for in-network video processing. For the best results, the use of domain specific information plus a level of flexible processing is required, which is not available in a router. In particular, Table 1 which has the characteristics of the video streams is used when processing chunks, such that if the network node deletes layer L1, then it also needs to delete layer L2. Routers cannot do this process.

In video streaming applications, a frame can also be considered as nominally lost and will be disregarded if it arrives late to the client, that is at a time later than the frame playout time. The results we have presented show that BPP provides the client with suitable video layers showing low outage at the available bandwidth. One of the important observations, over all the tests, is that all the frames arrived to the client at the appropriate time, so that none of the frames had to be disregarded. If traditional transmission is used, some frames will be lost whereas some others might arrive to the client much later than its deadline. This shows us that BPP is beneficial for the timing requirements of the frames, and that the approach used ensured the video packets are successfully transferred.

We presented the mechanisms and techniques utilized to take multi-layer SVC H264 video, and created suitable packing strategies and a packet structure amenable with BPP. We have shown that different packing strategies have a different effect on QoE, for the same available bandwidth. When the bandwidth values change, this effect changes, and we have determined that different packing strategies might be more advantageous than the others for these different bandwidth values. The QoE parameters observed for different packing strategies might help engineers design a video streaming system in which the server selects the packing strategy by considering the available bandwidth conditions.

The results also show that it is possible to provide even more improvement of the network resource utilization for such a system. Since SVC layers are dependent on each other, if an enhancement layer of a frame is removed by BPP during transmission, then it is not necessary to transmit upper enhancement layers of that frame. Also, some layers of some frames need to be divided into more than one packet due to their huge size, or the chunk arrangement within the packets. If a chunk of a layer is removed, then the remaining chunks of that layer should also be removed, as well as all the chunks of the upper layers of that frame. However, keeping track of the deleted chunks and layer inter-dependencies requires an additional mechanism, since it would not be reasonable to make BPP-enabled switches stateful. SDN is a good alternative which can be used as this additional mechanism. This is open to further investigation. We can implement an algorithm, which will run on an SDN controlled in-network compute server, to determine the chunks to be removed, not only by considering their significance values, also by considering the previous deletions and dependencies between the chunks and layers.

The observed effects of the packing strategies are initial results, and this acts as a foundation for further enhancements. In future work, we plan to dynamically change the network conditions, to evaluate the different packing strategies and their tradeoffs, by considering which one works best as the bandwidth varies dynamically over time. We plan to enhance the server, so it will get network related information from the client feedback, creating a video streaming system in which the server automatically selects the packing strategy based on the available bandwidth conditions. Also, enhancing the network function to know which packing strategy is used is planned, so it can wash away chunks in a way that is more effective for the video. We also plan further evaluations, to calculate the VMAF metric for the videos received using BPP, and to compare BPP based video streaming with HAS. Another aspect is to investigate the use of RTP content carried inside BPP chunks, to determine if these are complementary. As BPP and video streaming work well together, we believe the mechanisms presented here will be suitable for other low latency, high bandwidth domains such as online gaming, Augmented Reality (AR), and Virtual Reality (VR).

7 Open Source

For reproducibility we have provided instructions and the software.

The software for processing the H264 streams, the sender and the client, are here: https://github.com/stuartclayman/h264_over_bpp.

The VLSP Virtual Infrastructure environment is available here:

https://github.com/stuartclayman/VLSP.

The instructions plus the video utilized for the experiments, can be found here: https://github.com/stuartclayman/h264_over_bpp_via_vlsp.

Author Contributions Stuart Clayman and Müge Sayıt wrote the manuscript, prepared the figures and tables, and reviewed and editing the document in a shared and collaborative fashion.

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

Declarations

Conflict of interest We declare that the authors have no competing interests as defined by Springer, or other interests that might be perceived to influence the results and/or discussion reported in this paper.

References

1. Belshe, M., Peon, R., Thomson, M.: Hypertext Transfer Protocol Version 2 (HTTP/2). RFC 7540

2. Bentaleb, A., Taani, B., Begen, A.C., Timmerer, C., Zimmermann, R.: A survey on bitrate adaptation schemes for streaming media over HTTP. IEEE Commun. Surveys Tutorials **21**(1), 562–585 (2019)
3. Bishop, M.: Hypertext Transfer Protocol Version 3 (HTTP/3). Internet-Draft draft-ietf-quic-http-34, Internet Engineering Task Force (February 2021)
4. Clark, D.D., Tennenhouse, D.L.: Architectural Considerations for a New Generation of Protocols. In: SIGCOMM, Philadelphia, ACM (1990) 200–208
5. Clayman, S.: The Inter-Dependence of Network Transport and Application Behaviour. In: ITU Network 2030, Geneva (October 2019)
6. Clayman, S., Sayit, M.: In-network scalable video adaption using big packet protocol. In: Proceedings of the 12th ACM Multimedia Systems Conference. MMSys '21, pp. 363–368, ACM, New York (2021)
7. Clayman, S., Sayıt, M.: 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)
8. Clayman, S., Mamatas, L., Galis, A.: Experimenting with Control Operations in Software-Defined Infrastructures. In: 2016 IEEE NetSoft Conference and Workshops (NetSoft), pp. 390–396 (2016)
9. Clayman, S., Mamatas, L., Galis, A.: Efficient management solutions for software-defined infrastructures. In: NOMS 2016—2016 IEEE/IFIP Network Operations and Management Symposium, pp. 1291–1296 (2016)
10. Clayman, S., Tuker, M., Arasan, H., Sayıt, M.: 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)
11. Clayman, S., Tuker, M., Arasan, H., Sayıt, M.: Managing video processing and delivery using big packet protocol with SDN controllers. In: IEEE Conf. on Network Softwarization—Netsoft, Tokyo (July 2021)
12. Clemm, A., Eckert, T.: High-precision latency forwarding over packet-programmable networks. In: NOMS 2020—2020 IEEE/IFIP Network Operations and Management Symposium, pp. 1–8 (2020)
13. Crowcroft, J., Handley, M., Wakeman, I.: Internetworking Multimedia. CRC Press, Boca Raton (1999)
14. Dong, L., Li, R.: Distributed mechanism for computation offloading task routing in mobile edge cloud network. In: International Conference on Computing, Networking and Communications (ICNC), Honolulu, HI, USA, pp. 630–636 (2019)
15. Dong, L., Li, R.: In-packet network coding for effective packet wash and packet enrichment. In: 2019 IEEE Globecom Workshops (GC Workshops), pp. 1–6 (2019)
16. Durak, K., Akcay, M.N., Erinc, Y.K., Pekel, B., Begen, A.C.: Evaluating the performance of Apple's low-latency hls. In: 2020 IEEE 22nd International Workshop on Multimedia Signal Processing (MMSP), pp. 1–6 (2020)
17. Grafl, M., Timmerer, C., Hellwagner, H., Chérif, W., Ksentini, A.: Evaluation of hybrid scalable video coding for HTTP-based adaptive media streaming with high-definition content. In: 2013 IEEE 14th International Symposium on "A World of Wireless, Mobile and Multimedia Networks" (WoWMoM) (2013)
18. Huysegems, R., van der Hooft, J., Bostoen, T., Rondao Alface, P., Petrangeli, S., Wauters, T., De Turck, F.: HTTP/2-based methods to improve the live experience of adaptive streaming. In: Proceedings of the 23rd ACM International Conference on Multimedia. MM '15, New York, NY, USA, pp. 541–550. Association for Computing Machinery (2015)
19. Ibrahim, S., Zahran, A.H., Ismail, M.H.: SVC-DASH-M: Scalable Video Coding Dynamic Adaptive Streaming Over Http Using Multiple Connections. 2014 21st International Conference on Telecommunications (ICT), pp. 400–404 (2014)
20. Li, R., Clemm, A., Chunduri, U., Dong, L., Makhijani, K.: A new framework and protocol for future networking applications. In: NEAT 2018—Proceedings of ACM Workshop on Networking for Emerging Applications and Technologies (August 2018)
21. Li, R., et al.: A Framework for Qualitative Communications Using Big Packet Protocol. In: NEAT 2019: Proceedings of ACM Workshop on Networking for Emerging Applications and Technologies, pp. 22–28 (August 2019)
22. Lim, M., Akcay, M.N., Bentaleb, A., Begen, A.C., Zimmermann, R.: When they go high, we go low: low-latency live streaming in dash.Js with LoL. In: Proceedings of the 11th ACM Multimedia

Systems Conference. MMSys ’20, New York, NY, USA, pp. 321–326. Association for Computing Machinery, New York (2020)
23. Makhijani, K., Li, R., Boukary, H.E.: Using big packet protocol framework to support low latency based large scale networks. In: ICNS 2019, Athens (2019)
24. MPEG: MPEG-DASH: Dynamic Adaptive Streaming over HTTP
25. Palmer, M., Krüger, T., Chandrasekaran, B., Feldmann, A.: The QUIC fix for optimal video streaming. In: Proceedings of the Workshop on the Evolution, Performance, and Interoperability of QUIC. EPIQ’18, pp. 43–49. ACM, New York (2018)
26. Schulzrinne, H.: Real Time Streaming Protocol (RTSP). RFC 2326
27. Schulzrinne, H.: RTP: a transport protocol for real-time applications. RFC 3550
28. Schwarz, H., Marpe, D., Wiegand, T.: Overview of the scalable video coding extension of the H.264/AVC standard. IEEE Trans. Circuits Syst. Video Technol. **17**(9), 1103–1120 (2007)
29. Seufert, M., Egger, S., Slanina, M., Zinner, T., Hoßfeld, T., Tran-Gia, P.: A survey on quality of experience of HTTP adaptive streaming. IEEE Commun. Surveys Tutorials **17**(1), 469–492 (2015)
30. Seufert, M., Schatz, R., Wehner, N., Casas, P.: 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), pp. 7–12 (2019)
31. Soret, B., Mogensen, P., Pedersen, K.I., Aguayo-Torres, M.C.: Fundamental tradeoffs among reliability, latency and throughput in cellular networks. In: 2014 IEEE Globecom Workshops (GC Workshops), pp. 1391–1396 (2014)
32. Yadav, P.K., Bentaleb, A., Lim, M., Huang, J., Ooi, W.T., Zimmermann, R.: Playing chunk-transferred DASH segments at low latency with QLive. In: Proceedings of the 12th ACM Multimedia Systems Conference. MMSys ’21, New York, NY, USA. Association for Computing Machinery, pp. 51–64 (2021)
33. Yahia, M.B., Louedec, Y.L., Simon, G., Nuaymi, L., Corbillon, X.: HTTP/2-based frame discarding for low-latency adaptive video streaming. ACM Trans. Multimedia Comput. Commun. Appl. **15**(1), 1–23 (2019)
34. Yang, S., Wei, Z., Ling, L.: Multimedia communication and scalable video coding. In: 4th Intl Conf on Intelligent Computation Technology and Automation, Shenzhen, China, pp. 616–619 (2011)
35. Zhu, B., et al.: A real-time H.266/VVC software decoder. In: Proceedings of IEEE International Conference on Multimedia and Expo (ICME) (July 2021)

Stuart Clayman received his PhD in Computer Science from University College London in 1994. He is currently a Principal Research Fellow at UCL EEE department, and he has worked as a Research Lecturer at UCL and at Kingston University. He co-authored over 75 conference and journal papers. His research interests and expertise lie in the areas of software engineering and programming paradigms; distributed systems; virtualised compute and network systems, network and systems management; sensor systems and smart city platforms, and artificial intelligence systems. He is looking at enhanced mechanisms for end-to-end delivery of digital video, as well as new techniques for large-scale sensor systems in Industry 4.0. He also has extensive experience in the commercial arena undertaking architecture and development for software engineering, distributed systems, and networking systems. He has run his own technology start-up in the area of NoSQL databases, sensor data, and digital media.

Müge Sayıt is an associate professor at the International Computer Institute at Ege University in Turkey. She received her M.Sc. degree in 2005 and a Ph.D. degree in 2011 in Information Technologies from the same institute. Her research interests include Software Defined Networking, Network Function Virtualization, future networks, video streaming, and video codecs. She is a member of the IEEE P1916.1 Standard for Software Defined Networking and Network Function Virtualization Performance group. She has been working as the principal investigator or as a researcher in various R&D projects.