Analysis of H.264 Bitstream Prioritization for Dual TCP/UDP Streaming of HD Video Over WLANs

Mohammed Sinky, Arul Dhamodaran, and Ben Lee
School of Electrical Engineering and Computer Science
Oregon State University
Corvallis, OR 97331
Email: {sinky, dhamodaa, benl}@eecs.orst.edu

Jing Zhao
QualComm Inc.
WT-330D, 5745 Pacific Center Blvd,
San Diego, CA 92121 (USA)
Email: zpeter@qti.qualcomm.com

***Abstract*—Flexible Dual-TCP/UDP Streaming Protocol (FDSP) is a new method for streaming H.264-encoded HD video over wireless networks. The method takes advantage of the hierarchical structure of H.264/AVC syntax and uses TCP to transmit important syntax elements of H.264/AVC video and UDP to transmit non-important elements. FDSP was shown to outperform pure-UDP streaming in visual quality and pure-TCP streaming in delay. In this work, FDSP is expanded to include a new parameter called *Bitstream Prioritization* (BP). The newly modified algorithm, FDSP-BP, is analyzed to measure the impact of BP on the quality of streaming for partially and fully congested networks. Our analysis shows that FDSP-BP is superior to pure-TCP streaming methods with respect to rebuffering instances, while still maintaining high visual quality.**

I. Introduction

Peer-to-peer HD video streaming over WLANs is an important enabling technology for home-entertainment and N-screen applications. Some of the well-known products competing in this application domain are Apple AirPlay® [1], Chromecast® [2], and Intel WiDi® [3]. These products are able to operate with streaming servers for Video on Demand (VoD) as well as in peer-to-peer fashion for both interactive and non-interactive video. However, significant challenges exist in delivering smooth playback of HD content as this technology becomes more pervasive and multiple streams will need to be supported on the same network.

The three major interrelated factors that impact visual quality of videos streamed over WLANs are packet loss, packet delay, and the transport layer protocols used to transmit packets. UDP is more appropriate for the tight timing constraints of interactive video but suffers from packet loss. TCP guarantees packet delivery ensuring perfect video frame quality, but suffers from packet delay. Packet loss causes degradation in video quality while packet delay requires buffering leading to freezes in video playback.

Although TCP and UDP both have advantages and disadvantages, little attention has been paid to leveraging the benefits of both protocols to improve performance of HD video streaming in WLANs. Our prior work introduced a new wireless video streaming technique called *Flexible Dual-TCP/UDP Streaming Protocol* (FDSP) [4], which exploited the advantages of both TCP and UDP. FDSP works with H.264-encoded video and relies on TCP to transport important syntax elements of the coded bitstream, specifically *slice headers* and *parameter sets*. Guaranteed delivery of these important syntax elements gives any H.264-compliant decoder a better opportunity to successfully decode received video even when packet loss occurs. On the other hand, all other video data is transported over UDP. FDSP was shown to achieve higher PSNR than pure-UDP and less buffering time than pure-TCP, thus, striking a balance between visual quality and delay [4].

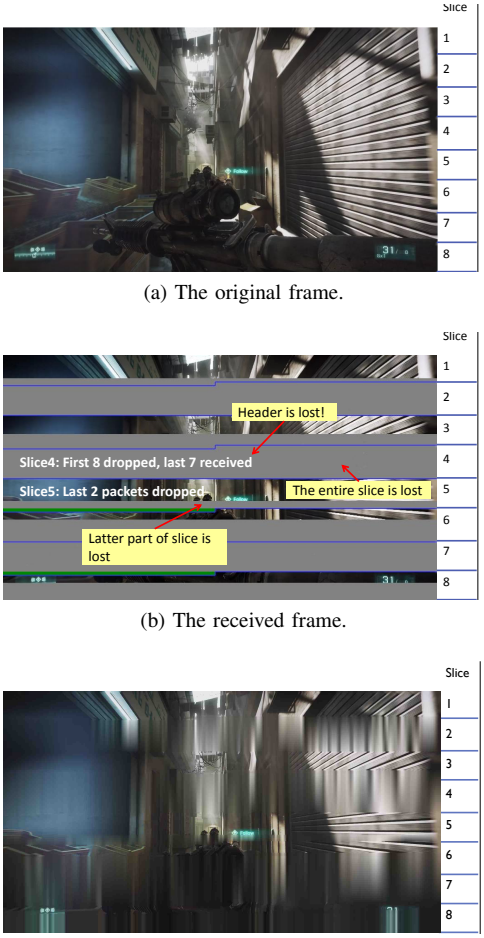
This paper proposes a new feature for the FDSP protocol, called *Bitstream Prioritization* (BP), which allows additional high-priority data to be sent over TCP. Most compressed video, including H.264, exhibits an inherent form of data priority related to frame types. *Intra-predicted* frames, e.g., I-frames, constitute high priority data within a compressed bitstream. Loss of I-frame data causes significant reduction in visual quality for subsequent *Inter-predicted* frames, e.g., P- and B-frames, because of their dependence on properly received I-frames. Thus, the newly proposed approach, FDSP-BP, targets a percentage of packets of I-frames to guarantee their delivery over TCP, which allows for better reconstruction of H.264-encoded video sequences.

Our simulation study shows that FDSP-BP significantly improves video quality as the BP percentage increases. Moreover, FDSP-BP always provides better video quality than pure-UDP streaming and significantly lower rebuffering times and rebuffering instances than pure-TCP streaming.

II. Background

This section provides a brief overview of the H.264 bitstream as well as a discussion on the impact of packet loss and delay on video quality. The basic structure of the H.264 bitstream consists of a series of Network Abstraction Layer (NAL) units such as Sequence Parameter Set (SPS), Picture Parameter Set (PPS), and slices. The SPS contains parameters that apply to the entire video sequence, such as the profile and level the coded video conforms to. Hence, loss of SPS would result in the loss of the entire video sequence. The PPS contains parameters that apply to the decoding of a sequence of coded frames, such as the employed entropy coding mode. If the PPS of a coded video sequence is lost, then those specific frames cannot be decoded.

In H.264, a slice is a basic spatial segment that is independent of its neighboring slices within a video frame. A frame can either have a single slice or multiple slices. Each slice can further be classified as an I/IDR-slice, P-slice, or B-slice depending on the type of frame that they belong to. A slice can further be classified into slice header and slice data. A *slice header* contains information that is common to all macroblocks (MBs) within a slice, and *slice data* refers to the actual MBs. Each slice can be subdivided into one or more packets for streaming. Thus, if a packet containing the slice header is lost, then that entire slice cannot be decoded even if all the MBs of slice data are received properly [5], [6]. This is illustrated in Fig. 1, where Fig. 1a is the original frame and Fig. 1b shows the received frame with some information missing due to packet loss. In Slice 4, the first 8 packets of the slice are lost, which include the slice header, and thus renders the entire slice undecodable. However, the decoder is able to reconstruct a large portion of Slice 5, whose slice header and most of the subsequent packets are received in tact. Typically, a decoder will attempt to mask the visual gaps using Error Concealment (EC) techniques, such as Weighted Pixel Average (WPA), Frame Copy (FC) and other methods, leading to a better presentation. Fig. 1c shows the EC techniques employed by the decoder to improve video quality.



(a) The original frame.

(b) The received frame.

(c) The received frame with Error Concealment (EC).

Fig. 1. Effect of packet vs slice header loss.

III. Related Work

UDP is generally accepted to be more suitable than TCP for real-time video streaming since it offers low end-to-end delay for video playout [5], [7]. UDP performance is further improved by employing EC techniques to reduce the impact of data loss [8], [9]. However, if important data, such as SPS, PPS, and slice headers are lost, the decoder simply cannot reconstruct the video even with the aid of EC. UDP packet loss is tolerated by employing Unequal Error Protection (UEP), which prioritizes important data [5], [10], [11]. More advanced UEP methods incorporate Forward Error Correction (FEC) [10], [11]. These methods are orthogonal to the proposed FDSP-BP, and thus, they can be used together.

Another approach to improve wireless video streaming is bitstream prioritization to ensure proper delivery of important data to the destination. In [12], a cross-layer packetization and retransmission strategy is proposed, where a video bitstream is prioritized based on distortion impact, delay constraints, and changing channel conditions. However, these parameters are heavily dependent on accurate feedback, which is bandwidth taxing. A modified slicing scheme that provides in-frame packet prioritization is proposed in [13], which exploits the unequal importance of different regions within a frame. These prioritization techniques, however, do not consider slice headers, SPS and PPS information for prioritization and hence are prone to slice and frame losses. Furthermore, authors in [12] do not consider H.264 videos while authors in [13] employ custom modifications to H.264 slicing making it unsuitable for any H.264-encoded videos.

This paper expands the scope of our prior research on the FDSP streaming protocol [4] to include bitstream prioritization (BP). The newly proposed method, FDSP-BP, allows further prioritization of an H.264 bitstream by specifying percentages of I-frames as a parameter to further protect visual quality. FDSP-BP has several advantages over existing methods. First, it operates on any H.264-encoded video without the need for modifications to the encoding scheme. Second, it offers flexibility for future enhancements to bitstream elements of the H.264 codec that may be prioritized, such as data partitions and slice groups. Third, usage of both UDP and TCP allow FDSP-BP to operate for real-time applications as well as stored video streaming.

IV. FDSP Overview

FDSP was proposed as a new network streaming method for HD H.264 content [4]. For the sake of completeness, this section provides a brief overview of the underlying details of FDSP. An important aspect of FDSP is that it was developed with H.264 video reconstruction in mind. The primary aim of FDSP is to give decoders the best chance for frame reconstruction by protecting slice headers using reliable transmission.

Fig. 2 shows the architecture of FDSP with the added Bitstream Prioritization (BP) input and modified MUX mod-

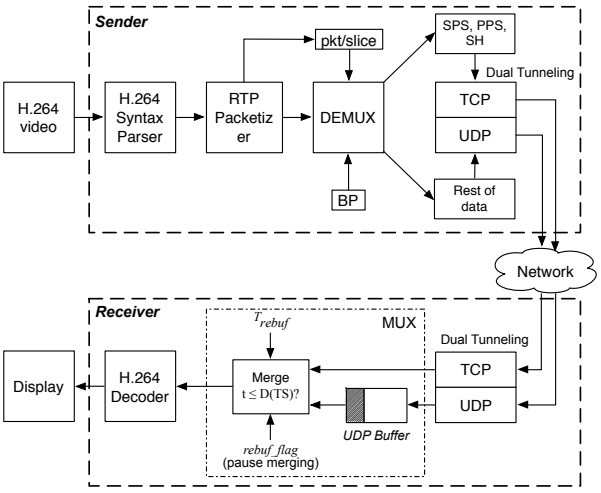


Fig. 2. Flexible Dual-tunnel Streaming Protocol (FDSP) Architecture [4] augmented with modified MUX and DEMUX modules for FDSP-BP.

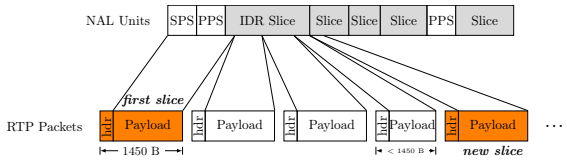


Fig. 3. IETF RFC 6184 RTP packetization of H.264 NAL Units modified to allow parameter set NAL Units to be grouped with VCL NAL Units (slices). RTP packets that hold H.264 slice headers are shown in orange.

ule. The four main components of the sender are (1) H.264 Syntax Parser, (2) RTP Packetizer, (3) DEMUX, and (4) Dual Tunneling (UDP+TCP). The H.264 Syntax Parser is responsible for identifying critical NAL units of the bitstream (i.e., SPS, PPS, and slice headers) and works together with the RTP Packetizer, which applies a slightly modified variation of IETF RFC 6184 for the packetization scheme [14]. Details of the packetization approach are given in Fig. 3. Then, MUX uses the information provided by the H.264 Syntax Parser to steer the critical RTP packets containing SPS, PPS or slice headers to the TCP tunnel, whereas packets classified as less important are sent to the UDP tunnel. Dual Tunneling keeps both UDP and TCP sessions active during video streaming.

The receiver contains the following three modules: (1) Dual Tunneling, (2) MUX, and (3) an H.264 decoder. As Dual Tunneling receives TCP and UDP packets, MUX is responsible for discarding those UDP packets that arrive beyond their playout deadlines. If a UDP packet is on time, it is merged with TCP packets of the same timestamp to form a complete frame, which is then sent to the decoder. The entire streaming process works in substream segments, where each substream is set to 10 sec.

V. FDSP Bitstream Prioritization

In FDSP, the number of packets selected for transmission over TCP depends on the number of slices present in the encoded video. For instance, in a single-slice encoding, only one RTP packet per frame is DEMUXed to the TCP socket.

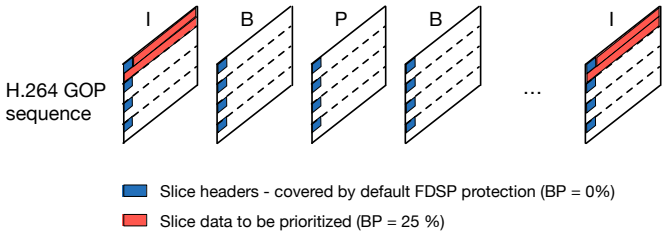


Fig. 4. BP applied to a 4-slice H.264 video sequence. When BP is sapplied, packets are selected sequentially from the start of the frame.

Based on the RTP packetization scheme shown in Fig. 3 with 10-second substreams and a frame rate of 30 fps, this amounts to sending only 1470 bytes of TCP data per frame (including TCP header), and results in a data rate of 0.35 Kbps. This data rate increases linearly with the number of slices. Thus, TCP data for a 4-slice video requires a throughput of 1.41 Kbps, which is still low considering a typical HD H.264 video generates an average bitrate of 4 Mbps. Our analysis shows that even in some bottleneck scenarios, TCP data is able to arrive well before its playout deadline.

The proposed FDSP-BP scheme takes advantage of the available slack time between when a substream is ready and its playout deadline to send additional high priority data over TCP. This is done by setting a percentage of packets for reference frames (i.e., I-frames and/or P-frames) that will be included for TCP transmission. A packet counter is added to the RTP Packetizer module to track the number of packets per frame, which is then fed to the DEMUX module. The BP value is provided as a second input in order to implement the new de-multiplexing technique for separation of data as shown in Fig. 2. Fig. 4 illustrates the application of the BP parameter to an I-frame. This means that in addition to sending SPSs, PPSs, and slice headers via TCP, a certain number of packets, counted from the start of an I-frame and within the percentage set by BP, will be sent over TCP. All other packets of the I-frame that do not fall under the category of a slice header/SPS/PPS or within the PBP percentage will be transmitted over UDP.

Although packets for both I-frames and P-frames can be included for TCP transmission, this paper analyzes the benefits of applying BP only on I-frames.

A. FDSP-BP Rebuffering

In the original implementation of FDSP [4], instances of rebuffering during video streaming did not take place. This is because only slice headers were targeted for prioritization, which represents a very small amount of data, and thus no deadline violations for TCP packets occurred. With the inclusion of BP, the chances of rebuffering are higher for FDSP-BP due to additional TCP data being streamed. To accommodate for this, the method by which incoming UDP packets are handled is modified within the MUX module.

For the default operation of FDSP, the portion of the first substream that is sent over TCP causes an initial delay, after

which UDP packet flow is initiated from the sender. Thus, an additional 150 ms of jitter time elapses at the receiver before playback begins. Then, incoming UDP packets are checked against their deadlines, D, according to the following equation:

$$D(TS) = T_{init} + T_{jitter} + TS \times \lambda, \quad (1)$$

where T_{init} is the initial buffering delay, T_{jitter} is the jitter time, TS is the RTP timestamp, and λ is the time interval between frames, $(1/fps)$. UDP packets are treated as real-time packets once transmission begins at T_{init}, and must adhere to the deadline D in Eq. 1. UDP packets arriving on or before their deadline, D, are merged with the corresponding TCP packets, while UDP packets arriving beyond D are discarded. However, TCP packets are treated much differently. *All* TCP packets for a given substream must be available for playback to continue; otherwise, video playback must be paused to allow for rebuffering.

When rebuffering occurs, Eq. 1 must be adjusted to accommodate for the duration that video is paused. Thus, FDSP-BP introduces two new inputs within the MUX module as shown in Fig. 2. In particular, the merging process now requires *rebuf_flag*, which is a flag that is raised when the TCP data for a given substream has not completely arrived by the corresponding playout deadline, and T_{rebuf}, which is the time spent during an instance of rebuffering. Whenever *rebuf_flag* is raised, merging of TCP and UDP data is suspended. Once all TCP data for the corresponding substream has arrived, the flag is de-asserted and merging resumes. Thus, for FDSP-BP, Eq. 1 is modified as follows to incorporate successive rebuffering instances:

$$D(TS)_i = T_{init} + T_{jitter} + TS \times \lambda + \sum_i T^i_{rebuf}, \quad (2)$$

where, i represents the rebuffering instance, and T^i_{rebuf} represents the time spent rebuffering for that instance.

VI. Experimental Setup

The primary video selected for our experiments consists of 4350 frames of full HD video (1920×1080 @30fps, 181 seconds) from "The Hobbit" movie trailer. The video is encoded using x264 with an average bit rate of 4 Mbps with four slices per frame. Our simulation environment is *Open Evaluation Framework For Multimedia Over Networks* (OEFMON) [15], which is composed of a multimedia framework, *DirectShow*, and a network simulator, *QualNet 5.0.2* [16]. OEFMON, which originally worked only with single-slice video, was further modified to allow for simulation of multi-slice H.264 video. Within OEFMON, an 802.11a ad-hoc network is setup with a bandwidth of 54 Mbps and the primary video described above is streamed using FDSP-BP. Three constant bitrate (CBR) background streams induce interference within carrier sense multiple access (CSMA) range of the primary video stream.

The simulation configuration is shown in Fig. 5, where the primary video described above is streamed between the pair of nodes 1 and 2. The remaining node pairs generate

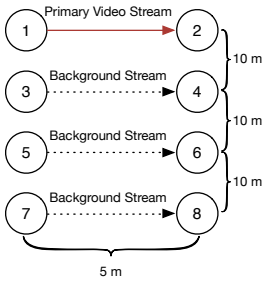


Fig. 5. Network Scenario.

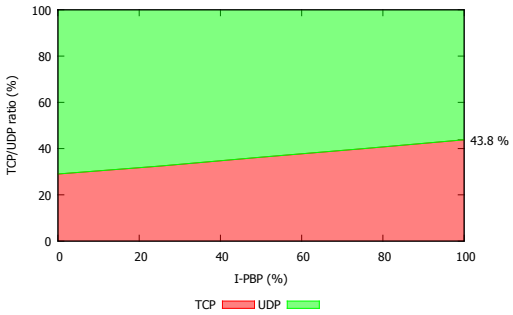


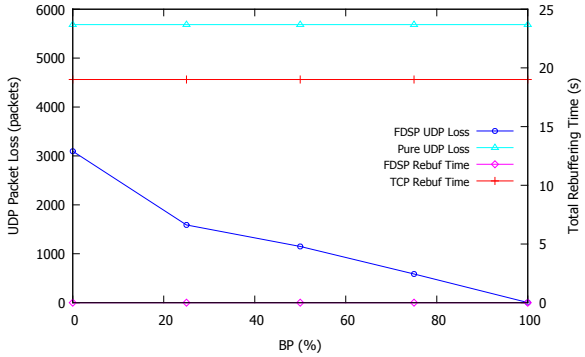
Fig. 6. The ratio of UDP and TCP packets for the test video as BP is increased from 0% to 100%.

background traffic with CBR data. Distances between each source and destination pair is 5 m and the distance between pairs of nodes is 10 m. These distances were chosen to mimic the proximity of multiple neighboring streaming devices in an apartment setting. Two network case studies are considered for evaluating FDSP-BP. The first case evaluates a partially congested network where the aggregate background traffic is 25 Mbps. The second case is a fully congested network where the aggregate background traffic is 50 Mbps. The test video packetized according to the discussion in Sec. IV consists of 60,044 RTP packets. Fig. 6 shows the ratio of total packets sent over TCP and UDP as BP is increased from 0% to 100% for the test video. For BP = 0%, which represents the default FDSP operation, nearly 30% of all packets are sent over TCP. In this case, all packets sent over TCP contain only slice headers with SPS and/or PPS information. When BP = 100%, entire I-frames are sent over TCP. For this case, approximately 44% of all packets are transmitted over TCP. It is worth noting that, in general, these ratios will depend on the number of slices per frame and the GOP sequence of videos.

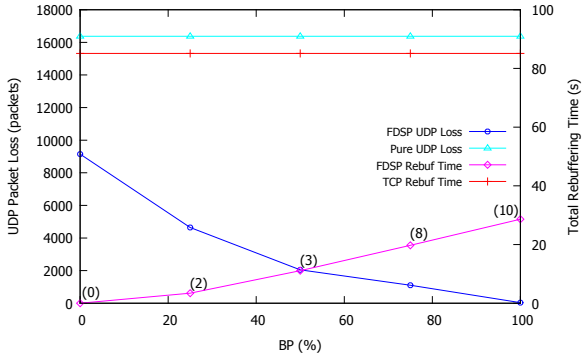
VII. Results

This section presents the performance results of FDSP-BP for partially and fully congested networks in comparison to pure-UDP and pure-TCP methods.

Fig. 7 compares the performance of pure-UDP, pure-TCP, and FDSP-BP in terms of packet loss and rebuffering time as a function of BP. Note that the UDP packet loss reported in the graphs considers both true packet loss and delayed



(a) Partially congested network with aggregate 25 Mbps CBR background traffic.



(b) Fully congested network with aggregate 50 Mbps CBR background traffic.

Fig. 7. Performance of pure-UDP, pure-TCP, and FDSP-BP for (a) partially congested network and (b) fully congested network scenarios with CBR background traffic. The numbers enclosed in parentheses on the graph for part (b) represent the number of rebuffering instances.

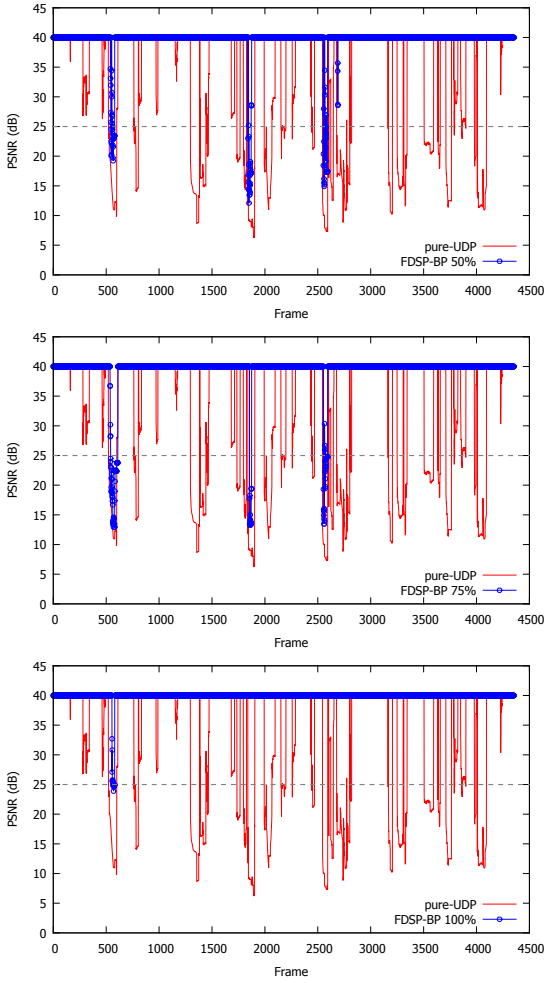


Fig. 8. PSNR results for the fully congested network case. Note the gradual improvement in PSNR as BP is increased. Improvements in PSNR result from less UDP packet loss. The grey dotted line at 25 dB represents the threshold for poor quality [17].

packets. True packet loss results when IP queue drops occur or when the MAC layer retransmission limit is exceeded. On the other hand, delayed packets are those UDP packets that were received but exceeded their playout deadlines.

Fig. 7a shows the results for the partially congested network scenario. The figure clearly shows that using FDSP-BP even without BP, i.e., BP = 0%, has an immediate impact on the number of lost packets compared to pure-UDP streaming. Comparison of *packet loss ratio* (PLR) for pure-UDP vs. FDSP-BP with BP = 0% yields a drop from 9.47% to 5.16%. BP values of 25%, 50%, 75%, and 100%, yield PLRs of 2.65 %, 1.92 %, 0.98%, and ∼0% (only 3 UDP packets are lost), respectively. In comparison to pure-TCP, FDSP-BP has no instances of rebuffering for all values of BP. In contrast, pure-TCP incurs four instances of rebuffering with a total rebuffering time of 19 sec., which amounts to 10% of the total video length.

Fig. 7b shows FDSP-BP performance for the fully congested network. This figure also includes the number of rebuffering instances (in parenthesis) encountered during FDSP-BP streaming. For example, no instances of rebuffering take place when BP = 0%. However, in contrast to the partially congested network case, rebuffering occurs as BP increases due to limited bandwidth availability leading to increased TCP delay. However, the total rebuffering time and the number of rebuffering instances are significantly less than that of pure-TCP based streaming. FDSP-BP incurs 28.6 sec. of rebuffering time with 10 instances of rebuffering when BP is 100%. With BP = 75%, there is 19.7 sec. of rebuffering time with 8 instances of rebuffering. With BP = 50%, there is 11.1 sec. of rebuffering time with 3 instances of rebuffering. In comparison, pure-TCP streaming results in 85.2 sec. of rebuffering time and 13 instances of rebuffering, which is four times higher than FDSP-BP with PB=50%.

Fig. 8 presents PSNR comparisons between FDSP-BP streaming and pure-UDP streaming for the fully congested network case. Note that when two frames are equivalent, its PSNR yields infinity. However, a PSNR value of 37 dB for a given frame is considered excellent quality [17]. Therefore, our PSNR results saturate at 40 dB, which for most practical purposes represents perfect frame reconstruction. Average PSNR values for FDSP-BP are 38.44 dB, 39.39 dB, and 39.92 dB

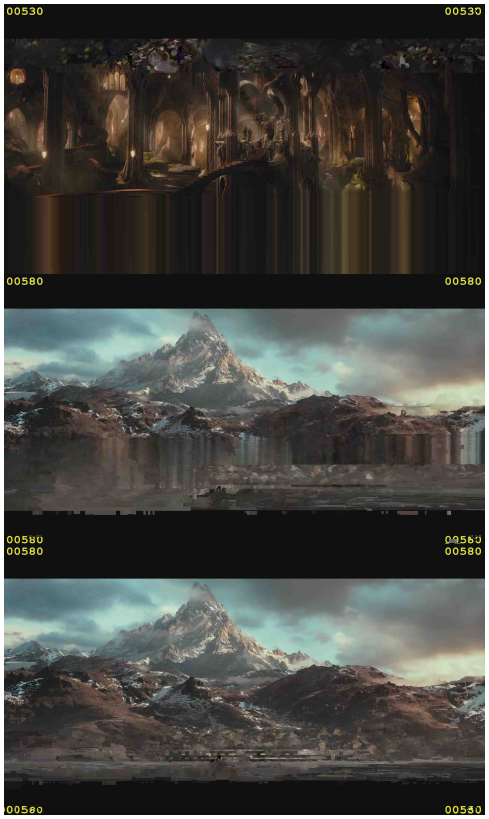


Fig. 9. The visual impact of increasing BP for the fully congested network case. The first screenshot is for pure-UDP streaming. Note that the frame is lost and duplication of a previously reconstructed frame (frame 530) is used. The second screenshot is for FDSP-BP streaming with BP = 50%, and the third screenshot is for FDSP-BP with BP =100%

with BP values of 50%, 75%, and 100%, respectively. On the other hand, pure-UDP streaming is very loss-prone and yields an average PSNR of 32.76 dB. These PSNR results show that FDSP-BP streaming does an excellent job of maintaining high perceived video quality.

Fig. 9 shows examples of frame quality improvement when BP is applied. These frames represent what is viewed by a user when frame 580 is expected to be displayed for pure-UDP streaming, FDSP-BP with BP = 50%, and FDSP-BP with BP = 100%. For the case of pure-UDP streaming (top image), 49 consecutive frames are lost (frames 531 to 580). Thus, for the duration of those frames, the last successfully decoded frame stored in the display buffer (i.e., frame 530) is continuously displayed, lasting for 1.63 seconds. For both cases of BP = 50% and 100%, since FDSP-BP guarantees slice header delivery through TCP, all four slice headers are properly received for frame 580. However, for BP = 50% (middle image), not all slice data is available due to UDP packet loss. As can be seen, information is missing for slices 3 and 4 leading to error concealment using Weighted Pixel Averaging (WPA). For the case of BP = 100%, degradation in the image is unnoticeable.

VIII. Conclusion

This paper introduced *Bitstream Prioritization* (BP) into the framework of the FDSP streaming protocol [4], i.e., FDSP-BP. FDSP-BP was evaluated for prioritizing packets of I-frames. An HD H.264-encoded video was streamed over a partially congested network and fully congested network to analyze the impact of the I-frame prioritization technique of FDSP-BP. Compared to the original implementation (i.e., BP = 0%), the new prioritization technique significantly improved video quality as the prioritization percentage was increased. Furthermore, significantly lower rebuffering times and instances were observed for FDSP-BP compared to pure-TCP streaming as well as notably better video quality than pure-UDP streaming.

As future work, FDSP-BP will be expanded into a bandwidth aware, dynamic streaming system where the BP percentage and sub-stream length are modified based on network conditions to minimize UDP packet loss and TCP retransmissions.

References

[1] "AirPlay." [Online]. Available: http://www.apple.com/itunes/airplay/
[2] "Chromecast." [Online]. Available: http://www.google.com/intl/en-us/chrome/devices/chromecast/
[3] "How to Connect Laptop to TV with Intel Wireless Display (WiDi)." [Online]. Available: http://www.intel.com/content/www/us/en/architecture-and-technology/intel-wireless-display.html
[4] J. Zhao, B. Lee, T.-W. Lee, C.-G. Kim, J.-K. Shin, and J. Cho, "Flexible dual tcp/udp streaming for h.264 hd video over wlans," in *Proc. of the 7th International Conference on Ubiquitous Information Management and Communication (ICUIMC '13)*. New York, NY, USA: ACM, 2013, pp. 34:1–34:9.
[5] S. Wenger, "H.264/AVC over IP," *IEEE Trans. Circuits Syst. Video Technol.*, vol. 13, no. 7, pp. 645–656, Jul. 2003.
[6] I. E. Richardson, *The H.264 Advanced Compression Standard*, 2nd ed. John Wiley and Sons, Ltd., 2010.
[7] D. Wu, Y. Hou, W. Zhu, Y.-Q. Zhang, and J. Peha, "Streaming video over the internet: approaches and directions," *IEEE Trans. Circuits Syst. Video Technol.*, vol. 11, no. 3, pp. 282–300, Mar. 2001.
[8] Y. Wang and Q.-F. Zhu, "Error control and concealment for video communication: a review," *Proc. IEEE*, vol. 86, no. 5, pp. 974–997, May 1998.
[9] Y. Xu and Y. Zhou, "H.264 video communication based refined error concealment schemes," *IEEE Trans. Consum. Electron.*, vol. 50, no. 4, pp. 1135–1141, Nov. 2004.
[10] A. Nafaa, T. Taleb, and L. Murphy, "Forward error correction strategies for media streaming over wireless networks," *IEEE Commun. Mag.*, vol. 46, no. 1, pp. 72–79, Jan. 2008.
[11] J. Kim, R. M. Mersereau, and Y. Altunbasak, "Distributed video streaming using multiple description coding and unequal error protection," *IEEE Trans. Image Process.*, vol. 14, no. 7, pp. 849–861, Jul. 2005.
[12] M. van der Schaar and D. Turaga, "Cross-layer packetization and retransmission strategies for delay-sensitive wireless multimedia transmission," *IEEE Trans. Multimedia*, vol. 9, no. 1, pp. 185–197, Jan. 2007.
[13] I. Ali, M. Fleury, S. Moiron, and M. Ghanbari, "Enhanced prioritization for video streaming over QoS-enabled wireless networks," in *Wireless Advanced (WiAd'11)*, Jun. 2011, pp. 268–272.
[14] Y.-K. Wang, R. Even, T. Kristensen, and R. Jesup, "RTP Payload Format for H.264 Video," RFC 6184 (Proposed Standard), Internet Engineering Task Force, May 2011.
[15] C. Lee, M. Kim, S. Hyun, S. Lee, B. Lee, and K. Lee, "OEFMON: An open evaluation framework for multimedia over networks," *IEEE Commun. Mag.*, vol. 49, no. 9, pp. 153–161, Sep. 2011.
[16] *QualNet 5.0.2 User's Guide*, Scalable Network Technologies, Inc., 2010.
[17] J. Gross, J. Klaue, H. Karl, and A. Wolisz, "Cross-layer optimization of OFDM transmission systems for mpeg-4 video streaming," *Computer Communications*, vol. 27, no. 11, pp. 1044 – 1055, Jul. 2004.