Analysis of Optimum Substream Lengths for Dual TCP/UDP Streaming of HD H.264 Video Over Congested WLANs

Arul Dhamodaran, Kevin Gatimu, and Ben Lee

School of Electrical Engineering and Computer Science
Oregon State University
Corvallis, Oregon 97331
Email: {dhamodar, gatimuk, benl}@eecs.orst.edu

***Abstract*—Flexible Dual-TCP/UDP Streaming Protocol (FDSP) is a new method for streaming H.264-encoded High-definition (HD) video over WLANs. FDSP streaming is done in sequential video segments or chunks called *substreams*. Substreams are typically used in HTTP/TCP streaming for efficient switching between different quality levels of video. However, in FDSP, substream lengths are used to control the amount of TCP data that needs to be sent prior to the playback of that substream. Substream length choice also affects the corresponding amount of UDP data. This paper presents an analysis of substream length modification in the context of FDSP. Our results show that as substream length increases, TCP rebuffering time decreases while the number of TCP rebuffering instances increases. However, our results also show that substream length alone does not have a clear correlation with UDP packet loss.**

***Keywords*—HD Video Streaming, Substream, Packet loss, Rebuffering, TCP, UDP**

I. Introduction

High-definition (HD) video streaming is a critical technology for today's multimedia applications. These video streaming applications can be broadly classified into Video on Demand (VoD) services and direct device-to-device steaming over WLANs. VoD services generally involve streaming servers that deliver HD video content to the end-user via the Internet, e.g., Netflix, Hulu, Amazon Video, etc. On the other hand, direct device-to-device streaming over WLANs is typically seen in home entertainment systems. They facilitate HD video sharing through screen mirroring and other N-Screen applications [1]. However, as more wireless devices become inter-connected, networks will need to support multiple video streams. This will pose significant challenges for providing uninterrupted HD video streaming.

Video streaming is based on either Transmission Control Protocol (TCP) or User Datagram Protocol (UDP). TCP facilitates HTTP-based streaming as is the case with VoD services. Meanwhile, direct device-to-device streaming applications that require real-time responsiveness rely on UDP. Using either UDP or TCP has its advantages and disadvantages. UDP is better suited for applications that require low latency, e.g., live video streaming, interactive video, and screen mirroring. However, UDP suffers from *packet loss* due to its unreliable nature, which results in degraded video quality. On the other hand, TCP is a reliable protocol that guarantees packet delivery, which ensures perfect video quality. However, TCP suffers from packet delay that leads to *rebuffering*, and thus stalled video playback.

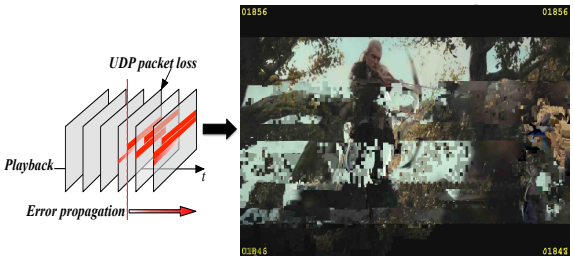


Fig. 1: Frame distortion due to UDP packet loss. Note that packet loss also causes frame distortion in subsequent frames due to error propagation.

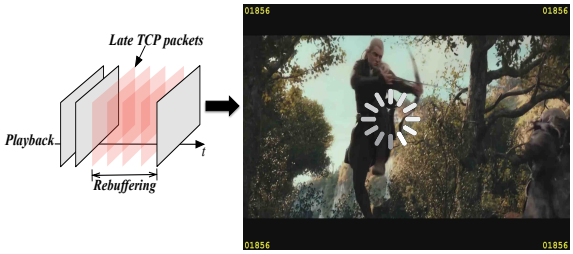


Fig. 2: Rebuffering due to late TCP packets.

Figure 1 illustrates the video quality degradation due to UDP packet loss, which affects not only the frame for which the packet loss occurred but also subsequent frames that use this frame as the reference frame (referred to as *error propagation*). Figure 2 illustrates the effect of rebuffering caused by TCP packets arriving at the receiver after the playout deadline. This delay causes the received video to stall and wait for more packets to arrive before resuming playback.

Our prior work in [2] proposed a new hybrid H.264 video streaming technique called *Flexible Dual-UDP/TCP Stream-*

ing Protocol (FDSP) that leverages the advantages of both UDP and TCP for HD video streaming between devices over a WLAN. FDSP uses TCP to guarantee delivery of the more important elements of a H.264-coded bitstream, such as Sequence Parameter Set (SPS), Picture Parameter Sets (PPSs), and slice headers, while the rest of the data is transported via UDP. This gives the H.264 decoder a better chance of decoding received video even when packet loss occurs. Overall, FDSP was shown to strike a balance between visual quality and delay by achieving higher video quality than pure-UDP and less rebuffering than pure-TCP. FDSP was improved upon in [3] by introducing *Bitstream Prioritization* (BP) to reduce UDP packet loss and error propagation. BP is an adjustable metric that is used to select additional H.264 data to be transported over TCP. Increasing BP decreased UDP packet loss but also increased TCP rebuffering time (i.e., the total amount of time the video spends in the stalled state) and instances (i.e., the number of times the video stalls). FDSP-BP was further improved by using *Adaptive BP* that dynamically adjusts the BP parameter based on estimated TCP rebuffering time and UDP packet loss ratio (PLR) [4]. This further reduced both packet loss and rebuffering.

FDSP-based streaming is done in sequential video chunks called *substreams*. For each substream, the important syntax elements are sent first via TCP and then the rest of the data is sent via UDP. Therefore, the substream length determines the amount of TCP data that needs to be sent prior to the playback of that substream. In addition, the amount of TCP data affects rebuffering time (i.e., delay) between substreams. It also determines TCP transmission time, which in turn affects whether or not UDP packets meet the playout deadline at the receiver. The substream length also affects the total number of substreams, which is directly related to the number of possible rebuffering instances. In our prior work on FDSP, the substream length was chosen to be fixed at 10 seconds [5]. This paper analyzes how varying substream lengths affect UDP PLR and TCP rebuffering time and instances. Based on our results, better insight can be developed on how FDSP-based streaming can be furthered enhanced.

II. Related Work

Substream implementations for video streaming applications are commonly found in HTTP Adaptive Streaming (HAS) technology [6]. HAS aims to adjust video to the best possible quality (i.e., bitrate) under fluctuating network conditions. As such, HAS segments a video into multiple substreams, and dynamically switches between different quality levels (i.e., bitrates) as the video transitions from one substream to the next based on network conditions.

Existing proprietary HAS technologies use different substream lengths. For example, Microsoft Smooth Streaming (MSS) specifies 2-second substreams [7], Apple HTTP Live Streaming (HLS) typically targets a maximum of 10 seconds per substream [8], and Adobe HTTP Dynamic Streaming (HDS) substreams vary between 2 and 5 seconds [9]. These substream lengths are relatively short so that the streaming system can react fast enough to changes in network conditions while avoiding extra overhead [10].

In addition to these proprietary solutions, Dynamic Adaptive Streaming over HTTP (DASH) is an open standard for HAS introduced by MPEG in 2012 [11]. DASH provides no specification on substream lengths, and instead, this is left to the implementer. For example, in [12], Scalable Video Coding (SVC) is used to provide different bitrate versions of a video in a single media file rather than using multiple files with varying quality levels. Since all versions are encoded in the same file, there is flexibility in choosing not only the quality level but also the substream length based on media content.

Another HAS improvement based on substream length was proposed in [5]. A substream length of around 10 seconds is found to be sufficient for calculating how quickly a receiver fetches a substream. The network bandwidth can be determined solely from the application layer by observing the difference between fetch time and playout duration for a particular substream, and used by the adaptation algorithm.

The aforementioned substream length implementations are designed for HTTP streaming, which is TCP-based. Furthermore, they are aimed at efficient switching between different quality levels of the same video in order to achieve the best quality of experience (QoE) [6]. As a result, they are orthogonal to FDSP-based streaming since it can be combined with HAS and used within each quality level. However, to the best of our knowledge, there is no prior work on substream lengths for video streaming that is based on both TCP and UDP. Therefore, this paper looks at how different substream lengths affect FDSP streaming. In addition, rather than optimizing bitrate switching, our analysis is focused on minimizing UDP packet loss and TCP rebuffering for a single version of high quality video.

III. Background on FDSP

This section provides an overview of FDSP, including its architectural features, video streaming using substreams, and factors affecting video quality. For more details, see [2], [3] and [4].

A. FDSP Architecture

FDSP is a new hybrid streaming protocol that combines the reliability of TCP with the low latency characteristics of UDP. Figure 3 shows the FDSP architecture consisting of a sender and a receiver.

At the sender, H.264 video data is first processed by the *H.264 Syntax Parser* in order to detect critical NAL units, i.e., SPS, PPS, and slice headers. The rest of the NAL units are primarily slice data. The *RTP Packetizer* then encapsulates each NAL unit according to the RTP Payload Format for H.264 video [13]. The *Demultiplexer* then directs the RTP packets containing critical NAL units to the TCP stream and the rest to the UDP stream. Meanwhile, *Dual Tunneling* keeps both TCP and UDP sessions active during video streaming. The *BP Selection* module sets the Bitstream Prioritization (BP) parameter, which represents a percentage of I-frame data

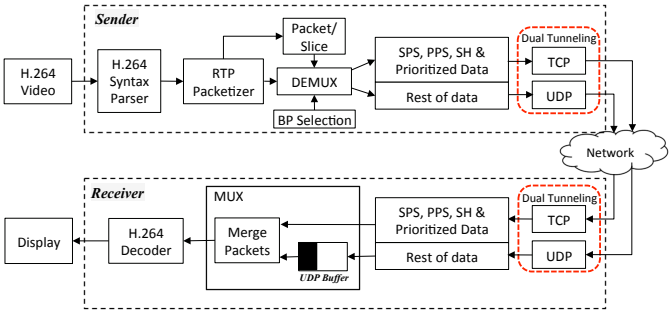


Fig. 3: Flexible Dual-UDP/TCP Streaming Protocol (FDSP) Architecture [2] augmented with modified MUX and DEMUX modules for FDSP-BP.

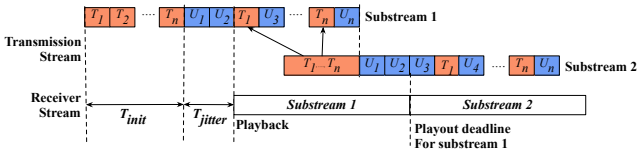


Fig. 4: Substream overlapping. Each packet is either UDP (U) or TCP (T), where the subscript represents the packet number within a substream.

that is also sent via TCP in addition to the original critical NAL units. BP is adjusted dynamically according to available network bandwidth.

In the receiver, the *Multiplexer* discards late UDP packets while rearranging TCP and UDP packets based on their RTP timestamps. This reordering is important for the *H.264 Decoder* to receive NAL units in the original decodable bitstream order.

B. FDSP Video Streaming using Substreams

During streaming, the FDSP sender subdivides the video into 10-second substreams. This minimizes initial buffering since all the TCP packets for a particular substream must be received before playback begins for that substream. To further minimize rebuffering, the TCP packets for the next substream are sent at the same time as the UDP packets for the current substream. This is called *substream overlapping* as illustrated in Figure 4. However, rebuffering occurs whenever TCP packets for the current substream are not yet all available and the receiver has to wait. The rebuffering time then postpones the playout deadline for all subsequent packets.

Substream overlapping is initiated only when the network device's IP queue is below the TCP threshold as shown in Figure 5. This prevents having too many TCP packets for the next substream in the queue, which would interfere with successful UDP packet transmission for the current substream (see Section III-C for more details on UDP packet loss). In [2], it was determined that 30% of the queue limit was the most suitable threshold.

C. Factors Influencing Video Quality in FDSP Streaming

FDSP-streamed video quality is affected by several factors, i.e., queue size, TCP threshold, estimated available bandwidth, BP parameter, and substream length. Since the main focus of this paper is to examine how different substream lengths affect FDSP video streaming, this subsection briefly discusses the

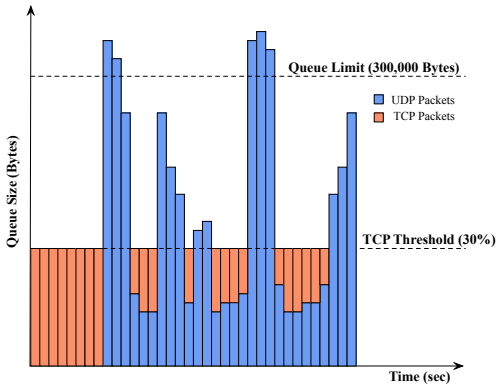


Fig. 5: IP queue containing both UDP and TCP packets. TCP packets are inserted when the queue size is below 30% of the limit (300 Kbytes). UDP packets inserted beyond the limit are considered lost.

other four factors and their impact on FDSP. These four factors are kept constant during our experiments (see Section IV for more details on the experiment setup).

1) Queue Size: This is the number of packets staged in the network device's IP queue prior to transmission. When the IP queue's limit is reached, any extra packets that are inserted are lost, which are referred to as queue drops. In FDSP, queue drops affect UDP packets, resulting in degraded video quality due to packet loss and error propagation.

2) TCP Threshold: This is the queue size threshold below which TCP packets can be inserted into the queue. The threshold is chosen such that TCP packets do not saturate the network due to retransmissions, which would lead to increased UDP packet loss and TCP rebuffering. Therefore, the TCP threshold helps mitigate these effects.

3) Estimated Available Bandwidth and BP: TCP round-trip time and UDP packet loss are used to estimate the available bandwidth, which determines how much data should be sent via TCP versus UDP based on the BP parameter. This further reduces UDP packet loss while keeping TCP rebuffering time and instances low.

IV. Experimental Setup

For our experiments, two full HD (1920×1080 @30fps) 3-minute clips from a high-motion (animation) video *Bunny*, and a low-motion (documentary) video *Bears* are used. These videos are 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) [14], which is composed of a multimedia framework, DirectShow, and a network simulator, QualNet 7.3 [15].

OEFMON, which originally worked with only single-slice video, was modified to allow for simulation of multi-slice H.264 videos. Within OEFMON, an 802.11g[1] ad-hoc network with a total bandwidth of 54 Mbps is set up as shown in Figure 6. The distance 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 pairs of neighboring streaming devices in an apartment setting. The

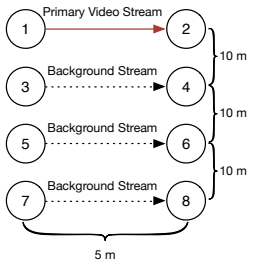


Fig. 6: Network configuration with 54 Mbps total bandwidth for simulating FDSP-based streaming with background traffic.

primary video is streamed between nodes 1 and 2. At the same time, the remaining three node pairs simulate either a partially or fully congested network by producing three constant bitrate (CBR) background streams that induce interference within Carrier Sense Multiple Access (CSMA) range of the primary 4 Mbps video stream. The partially congested configuration has a total 20 Mbps of background CBR traffic (i.e., 5, 10, and 10 Mbps streams), while the fully congested configuration has a total of 50 Mbps of background CBR traffic (i.e., 10, 20, and 20 Mbps streams).

The primary video is streamed using FDSP with BP of 0% and 100%. These two choices of BP values are based on our prior work [3], which showed that they represent the two extreme effects of FDSP-based streaming, i.e., UDP PLR and TCP rebuffering are maximized at BP of 0% and 100%, respectively, for 10-second substreams. Therefore, UDP PLR and TCP rebuffering are effectively separated via their corresponding BP values in order to study how substream length changes affect FDSP. For each BP value, 15 different substream lengths, ranging from 1 to 15 seconds, are evaluated for each video.

Finally, the IP queue size limit and the TCP threshold are set at 300 Kbytes and 30%, respectively, as shown in Figure 5. Most video streaming applications today use a dynamic queue size that is application specific [16]. For our simulations, a fixed queue size of 300 Kbytes is used.

V. Results

This section presents the results of our study on how substream length affects FDSP streaming in terms of TCP rebuffering and UDP PLR.

A. Impact of Substream Length on TCP Rebuffering

Figures 7 and 8 show the effects of substream length on the two test videos in both partially and fully congested networks, respectively, with BP of 100%.

For the partially congested network scenario, the number of FDSP rebuffering instances for both videos decreases slightly as the substream length increases, and then remains constant. For example, in the *Bunny* video (Figure 7a), substream lengths of 1 and 2 seconds incur 6 and 2 rebuffering instances with a total rebuffering time of 3.26 and 3.34 seconds, respectively. In the *Bears* video (Figure 8a), 1- and 2-second substreams incur 8 and 5 rebuffering instances with a total rebuffering times of 2.41 and 2.87 seconds, respectively. The results in Figures 7a and 8a also show that even with sufficient bandwidth, pure-TCP rebuffering still occurs and is greater than FDSP-TCP rebuffering. For example, the total pure-TCP rebuffering times for *Bunny* and *Bears* are 19.6 and 16 seconds, respectively.

In comparison to the partially congested network scenario, the rebuffering results for the fully congested network scenario are much more pronounced. For example, in the *Bunny* video (Figure 7b), there are 51 instances of rebuffering for 1-second substreams, compared to 8 instances for 15-second substreams. This is because longer substreams result in fewer substreams for a given video duration, and thus fewer substream transitions. Therefore, there are correspondingly fewer opportunities for the video to stall. Meanwhile, 1-second substreams have a total buffering time of 26.52 seconds compared to 42 seconds for 15-second substreams.

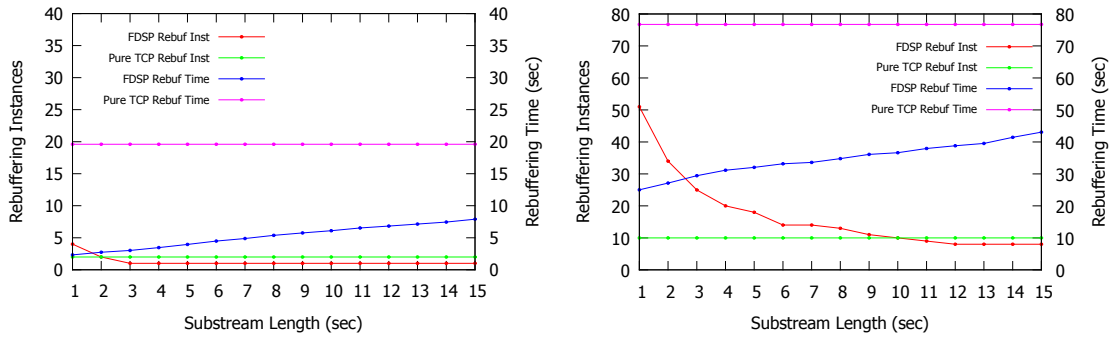
Figure 8b shows that the rebuffering characteristics of *Bears* are similar to those of *Bunny* for the fully congested network scenario. However, rebuffering in *Bears* is less pronounced than that in *Bunny* due to lower level of motion. The *Bunny* video has a significantly higher level of motion than *Bears*, and thus includes more TCP packets for I-frames [17]. This is especially the case for BP of 100%, where all I-frame data is sent through TCP.

Note that the overall number of TCP packets for a video at a particular BP value remains constant in spite of changes in substream length. However, a longer substream will result in a larger number of UDP packets in the IP queue, which causes the queue size to be larger than the TCP threshold for longer periods. Therefore, there are fewer opportunities to insert TCP packets into the IP queue. This means that less TCP packets are sent through substream overlapping and, instead, more are buffered in between substreams, thus increasing the total TCP rebuffering time. Furthermore, the network becomes more saturated as the IP queue fills up with more UDP packets. This increases the TCP average round-trip time and packet loss, and thus the number of retransmissions.

Nevertheless, the FDSP rebuffering times for both scenarios are less than pure-TCP rebuffering time, which is consistent with results from our previous work [3]. For this reason, the average time per instance of pure-TCP rebuffering is high despite the corresponding low number of rebuffering instances. For example, in the fully congested network scenario with *Bunny*, there are just 10 instances of pure-TCP rebuffering, but each one lasts an average of 7.8 seconds.

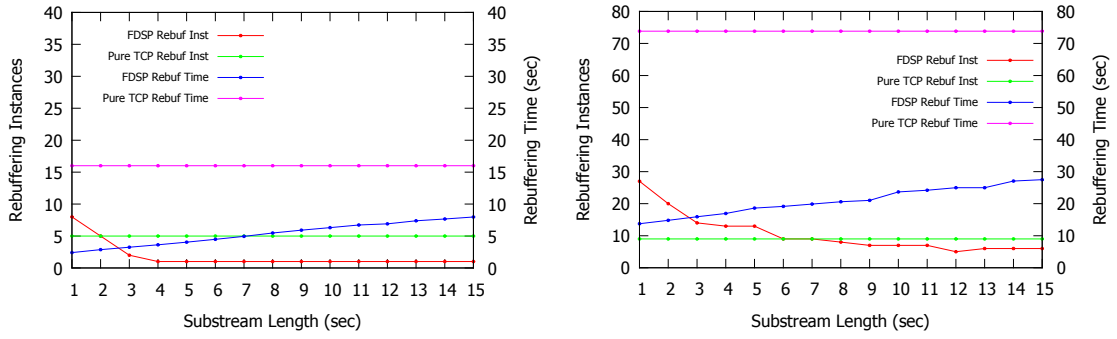
In comparison to BP of 100%, a BP value of 0% has minimal effect on TCP rebuffering, which is consistent with our prior work for fixed 10-second substreams [3], [4]. For example, in the fully congested network scenario with *Bunny*, only one instance of rebuffering occurs with a total rebuffering time of 1.54 seconds for 1-second substreams compared to 6.3 seconds for 15-second substreams.

[1]The version of the QualNet simulator used for our study only supports the IEEE 802.11g standard. However, the simulation study can be easily adopted to 802.11n and 802.11ac by having more background traffic to saturate the network.



(a) Partially congested network (b) Fully congested network

Fig. 7: Impact of substream length on TCP rebuffering time and instances in *Bunny* in partially and fully congested networks with BP of 100%.



(a) Partially congested network (b) Fully congested network

Fig. 8: Impact of substream length on TCP rebuffering time and instances in *Bears* in partially and fully congested networks with BP of 100%.

B. Impact of Substream Length on UDP Packet Loss

Figure 9 shows how substream length affects UDP packet loss with BP of 0% in partially and fully congested network scenarios for both videos. The partially congested network scenario has less PLR than the fully congested one due to higher bandwidth availability. Nonetheless, FDSP streaming in both scenarios exhibits erratic fluctuations in UDP PLR with respect to substream length changes.

Furthermore, there are multiple peaks in PLR at different substream lengths. This behavior can be attributed directly to peaks in TCP transmission time. For example, in Figure 9b, there are spikes in UDP PLR for substream lengths of 8, 11 and 13 seconds in the fully congested network scenario. Table I shows the relationship between the total TCP Transmission time and UDP PLR for these substream lengths (indicated in bold) as well as the adjacent ones. As can be seen, peaks in UDP PLR correspond to peaks in TCP transmission time. At the same time, TCP transmission time is affected by factors such as IP queue limit, average IP queue level, and TCP threshold. Analysis of these factors in the context of UDP packet loss is left as a future work. Nevertheless, UDP packet loss for FDSP at different substream lengths is always less than pure-UDP packet loss, which is consistent with results in our previous work [2].

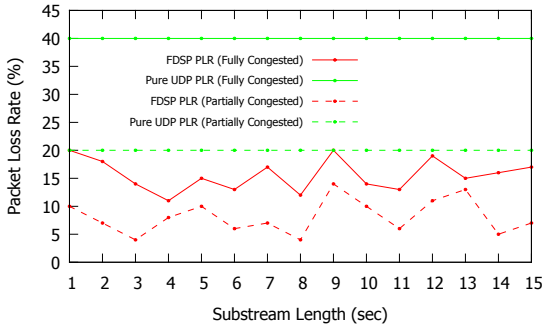
These results show that there is no significant correlation between substream length changes and UDP PLR. Our results also show that there is minimal UDP packet loss for BP of 100%, i.e., less than 1%. This is because the high ratio of TCP-to-UDP packets decreases the chances of UDP packet loss. This matches our results for fixed 10-second substreams [3], [4].

C. Recommendation on Substream Length Choice

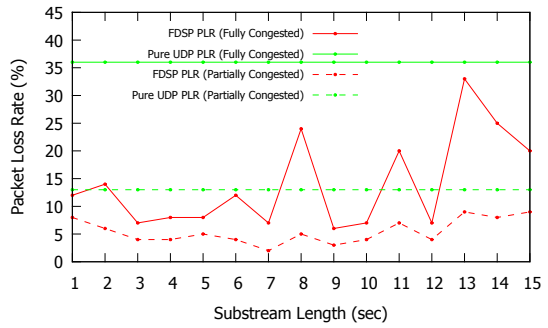
Our study shows that the most appropriate range of substream length is from 6 to 10 seconds. This is based on the fully congested network scenario because it affects TCP rebuffering and UDP PLR more adversely compared to the partially congested network scenario. The lower bound is chosen because the number of rebuffering instances increases almost exponentially for every 1-second decrease in substream length below 6 seconds (see Figures 7b and 8b). Furthermore, the number of rebuffering instances decreases up to the 12-second substream mark. Beyond this, there is no further improvement in rebuffering instances. However, the upper

Substream Length (sec)	Total TCP Tx Time (sec)	UDP PLR (%)
7	96.83	7
8	**99.64**	**25**
9	95.26	6
10	95.48	7
11	**98.96**	**20**
12	95.2	7
13	**100.12**	**33**
14	96.83	25

TABLE I: Relationship between the total TCP transmission time and UDP PLR in *Bears*. The bold entries correspond to spikes in UDP PLR.



(a) High-motion video (*Bunny* animation)



(b) Low-motion video (*Bears* documentary)

Fig. 9: Impact of substream length on UDP packet loss in partially and fully congested network scenarios with BP of 0%.

bound on substream length was chosen to be 10 seconds based on suitable initial buffering time. The 6-second and 10-second substream length videos have an initial buffering of about 5 seconds and 8 seconds, respectively, under the fully-congested network condition. However, in an ideal network condition, this initial delay is only 3 seconds and 5 seconds, respectively. These results show that initial delay incurred by FDSP streamed videos falls well within the user-acceptable range of 8 to 16 seconds [18].

Our recommendation for substream length choices does not take into account UDP PLR, which requires further analysis based on the results as discussed in Section V-B.

VI. Conclusion and Future Work

This paper analyzed the effect that different substream lengths have on video streaming in the context of Flexible Dual-TCP/UDP Streaming Protocol (FDSP). Our study showed that substream lengths have a direct effect on TCP rebuffering time and instances. As substream length increases, TCP rebuffering time decreases while the number of TCP rebuffering instances increases. However, substream length changes do not have a clear correlation with UDP packet loss. This lack of correlation is mainly attributed to its dependence on other parameters such as IP queue size, TCP threshold, etc. Our results show that a substream length of 6 to 10 seconds is best suited for FDSP video streaming.

As future work, we plan to analyze the effects of changing IP queue limit and TCP threshold together with varying substream length in order to further investigate FDSP-UDP packet loss. We also plan on modifying BP and substream length together towards developing a system with dynamic substream length modification.

References

[1] C. Yoon, T. Um, and H. Lee, "Classification of N-Screen Services and its standardization," in *2012 14th International Conference on Advanced Communication Technology (ICACT)*, Feb. 2012, pp. 597–602.

[2] 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 2013)*, Kota Kinabalu, Malaysia, 2013, pp. 34:1–34:9.

[3] M. Sinky, A. Dhamodaran, B. Lee, and J. Zhao, "Analysis of H.264 Bitstream Prioritization for Dual TCP/UDP Streaming of HD Video Over WLANs," in *IEEE 12th Consumer Communications and Networking Conference (CCNC 2015)*, Las Vegas, USA, Jan. 2015, pp. 576–581.

[4] A. Dhamodaran, M. Sinky, and B. Lee, "Adaptive Bitstream Prioritization for Dual TCP/UDP Streaming of HD Video," in *The Tenth International Conference on Systems and Networks Communications (ICSNC 2015)*, Barcelona, Spain, November 2015, pp. 35–40.

[5] C. Liu, I. Bouazizi, and M. Gabbouj, "Rate Adaptation for Adaptive HTTP Streaming," in *Proceedings of the Second Annual ACM Conference on Multimedia Systems*, ser. MMSys '11. New York, NY, USA: ACM, 2011, pp. 169–174. [Online]. Available: http://doi.acm.org/10.1145/1943552.1943575

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

[7] A. Zambelli. Smooth Streaming Technical Overview. [Online]. Available: http://www.iis.net/learn/media/on-demand-smooth-streaming/smooth-streaming-technical-overview

[8] Apple Inc. HTTP Live Streaming Internet—Draft. [Online]. Available: https://tools.ietf.org/html/draft-pantos-http-live-streaming-19

[9] Adobe Systems. HTTP Dynamic Streaming. [Online]. Available: http://www.adobe.com/products/hds-dynamic-streaming.html

[10] K. Ramkishor, T. S. Raghu, K. Siuman, and P. S. S. B. K. Gupta, "Adaptation of video encoders for improvement in quality," in *Proceedings of the 2003 International Symposium on Circuits and Systems, 2003. ISCAS '03*, vol. 2, May 2003, pp. II–692–II–695 vol.2.

[11] "ISO/IEC 23009-1:2012 - Information technology – Dynamic adaptive streaming over HTTP (DASH) – Part 1: Media presentation description and segment formats." [Online]. Available: http://www.iso.org/iso/iso_catalogue/catalogue_tc/catalogue_detail.htm?csnumber=57623

[12] T. Schierl, Y. S. d. l. Fuente, R. Globisch, C. Hellge, and T. Wiegand, "Priority-based Media Delivery using SVC with RTP and HTTP streaming," *Multimedia Tools and Applications*, vol. 55, no. 2, pp. 227–246, Sep. 2010. [Online]. Available: http://link.springer.com.ezproxy.proxy.library.oregonstate.edu/article/10.1007/s11042-010-0572-5

[13] 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. [Online]. Available: http://www.ietf.org/rfc/rfc6184.txt

[14] C. Lee, M. Kim, S. Hyun, S. Lee, B. Lee, and K. Lee, "OEFMON: An open evaluation framework for multimedia over networks," *IEEE Communications Magazine*, vol. 49, no. 9, pp. 153–161, Sep. 2011.

[15] *QualNet 7.3 User's Guide*, Scalable Network Technologies, Inc., 2016.

[16] S. Wei, G. Bai, and H. Shen, "An adaptive queue management mechanism for video streaming over mobile ad hoc networks," in *2009 5th International Conference on Wireless Communications, Networking and Mobile Computing*, Sept 2009, pp. 1–4.

[17] A. Divakaran, R. Radhakrishnan, and K. A. Peker, "Motion activity-based extraction of key-frames from video shots," in *2002 International Conference on Image Processing. 2002. Proceedings*, vol. 1, 2002, pp. I–932–I–935 vol.1.

[18] T. Hossfeld, S. Egger, R. Schatz, M. Fiedler, K. Masuch, and C. Lorentzen, "Initial delay vs. interruptions: Between the devil and the deep blue sea," in *2012 Fourth International Workshop on Quality of Multimedia Experience (QoMEX)*, Jul. 2012, pp. 1–6.