Adaptive Queue Management Scheme for Flexible Dual TCP/UDP Streaming Protocol

Arul Dhamodaran, Kevin Gatimu, and Ben Lee

School of Electrical 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 wireless networks. FDSP streaming is done in sequential video segments or chunks called substreams. In FDSP, substream lengths are used to control the amount of Transmission Control Protocol (TCP) data that needs to be sent prior to the playback of that substream. To avoid frequent rebuffering, TCP packets of the next substream are overlapped with the User Datagram Protocol (UDP) packets of the current substream. The TCP threshold parameter determines when to overlap new TCP packets with the current UDP stream. This paper analyzes the TCP threshold parameter in the context of FDSP. Our results show that user Quality of Experience (QoE) can be enhanced by adaptive adjusting of the TCP threshold using the additive-increase/multiplicative-decrease (AIMD) algorithm based on the UDP packet loss rate and the TCP rebuffering.**

Keywords–Bitstream Prioritization; HD Video Streaming; Queue Size; TCP Threshold; FDSP.

I. INTRODUCTION

High-definition (HD) video streaming technologies have fundamentally changed the way multimedia content is consumed. These video streaming applications can be broadly classified into client-server and device-to-device streaming applications. Client-server based streaming applications, such as Netflix, Hulu, Amazon Video, etc., typically involve a streaming server to deliver multimedia content to the end user through the internet. On the other hand, device-to-device wireless HD video streaming is enabled by technologies such as Apple AirPlay®, Google Chromecast®, and Wi-Fi Alliance's Miracast® to facilitate various multi-screen (i.e., N-screen) applications [1]. However, the explosion of wireless enabled devices will strain the bandwidth limits due to the need to support multiple streams in the same network.

All the aforementioned video streaming services and applications rely on either Transmission Control Protocol (TCP) or User Datagram Protocol (UDP) protocol. The client-server streaming techniques rely primarily on HTTP-based streaming, which, in-turn, is based on TCP. On the other hand, device-to-device streaming applications Chromecast and Miracast rely on UDP while Airplay uses TCP for video streaming and screen mirroring applications. However, both TCP- and UDP-based streaming protocols have their own set of challenges. TCP guarantees packet delivery ensuring perfect video frame quality, but suffers from freeze frames during video playback due to packet delay caused by bandwidth bottleneck. Figure 1 illustrates the effect of rebuffering caused by TCP packet delay, which occurs when TCP packets arrive at the receiver after the playout deadline due to network congestion. This delay causes

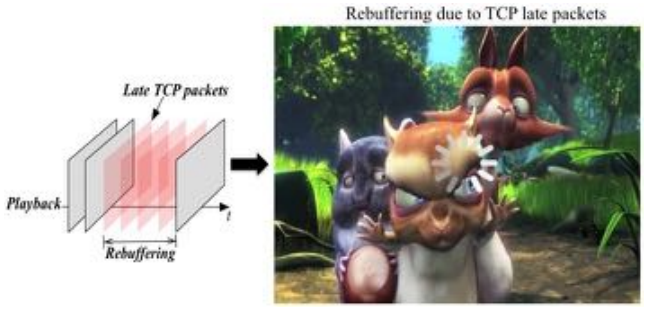


Figure 1. Rebuffering due to late TCP packets.

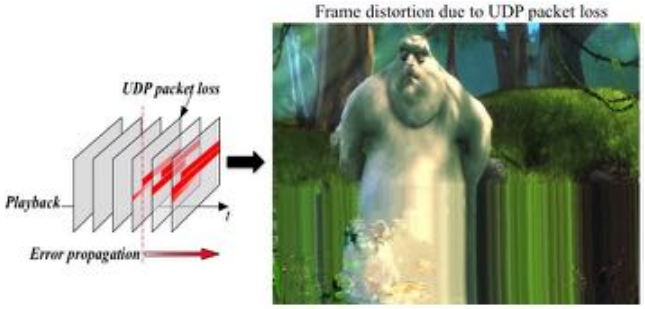


Figure 2. Frame distortion due to UDP packet loss. Note that packet loss also causes frame distortion in subsequent frames due to error propagation.

the received video to freeze frame and stall for more TCP packets to arrive at the receiver before resuming playback. UDP, on the other hand, minimizes delay but suffers from packet loss. Figure 2 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 it as the reference frame (referred to as *error propagation*).

In our previous work, a new H.264 based video streaming technique called *Flexible Dual Streaming Protocol* (FDSP) was proposed [2]. FDSP sends packets containing important H.264 video syntax elements (i.e., Sequence Parameter Set (SPS), Picture Parameter Set (PPS), and slice headers) via TCP for guaranteed delivery and the rest of the slice data packets via UDP giving an H.264 decoder a better chance of decoding received video even when packet losses occur. Therefore, FDSP exploits the combined benefits of TCP and UDP by adding reliability to UDP while reducing the latency caused by TCP. This enables FDSP 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 enhanced in [3] using *Bitstream Prioritization* (BP) to reduce the impact of UDP packet loss. This method

 ISBN: 978-1-61208-548-7

statically chooses the BP metric to classify a select percentage of originally UDP-designated packets from an H.264 bitstream as high priority, which are then transported over TCP for guaranteed delivery. FDSP-BP was further enhanced by introducing Adaptive-BP [4], where the percentage of packets sent over TCP versus UDP is dynamically adjusted based on the estimated rebuffering time for TCP packets and estimated packet loss ratio (PLR) for UDP packets. FDSP with Adaptive-BP further improved the performance by reducing both packet loss and rebuffering time.

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. To allow TCP packets to arrive on time, *substream overlapping* is performed where TCP packets for the next substream are sent at the same time as the UDP packets for the current substream. However, an important issue with substream overlapping is the decision on when to insert new TCP packets into the outgoing IP queue of the sender, which is referred to as *TCP Threshold*. In our prior work on FDSP, the TCP threshold was chosen to be fixed at 35% of the maximum IP Queue size [3]. This paper analyzes how varying the TCP threshold affects UDP PLR and TCP rebuffering time and develops an Adaptive TCP Threshold technique to improve user Quality of Experience (QoE).

This paper is organized as follows. Section II discusses other TCP and UDP streaming techniques. An overview of the FDSP streaming method is shown in Section III. Section IV discusses the effects of substream overlapping and TCP threshold. Sections V goes over the experimental setup and Section VI discusses the results of our analysis on how the changes in the TCP threshold affect FDSP streaming. Finally, Section VII concludes the paper.

II. Related Work

Queue management techniques for video streaming applications have been well studied. The two basic approaches proposed for queue management are Random Early Detection (RED) [5] and Blue [6]. Both of these techniques use queue length as an indicator of congestion and use this information to regulate the packet drop rates. Xu *et al.* proposed an active queue management technique for wireless ad hoc networks, called Neighborhood RED (NERD) [7]. This technique uses channel utilization to estimate the queue length to help determine the packet drop probability.

However, all the above queue management techniques are designed for data communication in general, without any consideration for the unique characteristics of video streaming (i.e., multimedia communication). Chen *et al.* proposed an active queue management technique where packets that may potentially be late are actively dropped before they are transmitted to reduce the strain on the network resources and to effectively control the transmission queue length [8]. Shy *et al.* proposed another active queue management (AQM) system, which employs routers that deal with both best-effort traffic flows and multimedia traffic flows [9]. Round trip time (RTT) is used in the packet dropping probability calculations to assure rate reductions in both multimedia and best-effort flows before the queue becomes full.

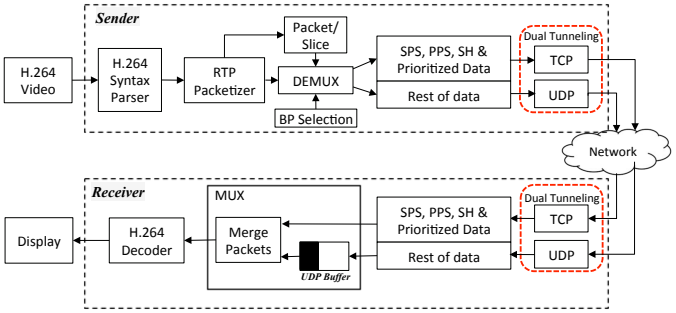


Figure 3. The architecture of FDSP with Adaptive BP [4]

The aforementioned queue management techniques are designed for video streaming systems that are based either on TCP or UDP. Furthermore, all of these systems focus on techniques such as prioritized dropping based on queue length, fairness based scheduling algorithms for packet delay optimization, etc. On the other hand, FDSP is a hybrid streaming protocol that uses both TCP and UDP for video streaming, and thus it presents a unique set of challenges, such as the TCP threshold, that play a crucial role in packet delay optimization. Therefore, this paper expands on the scope of our prior research on FDSP with Adaptive-BP [4] by analyzing the TCP threshold parameter and its impact on UDP PLR and TCP rebuffering.

III. FDSP Overview

FDSP was proposed as a new device-to-device video streaming technique for H.264 content [2]. This section provides a brief overview of its various architectural features and factors affecting video quality (see [2]-[4] for details).

FDSP with Adaptive-BP architecture is shown in Figure 3 [4], which consists of a sender and a receiver. The FDSP sender processes H.264 video data using the *H.264 Syntax Parser* to detect important Network Abstraction Layer (NAL) units, i.e., SPS, PPS, and slice headers (SH). The rest of the NAL units are primarily slice data packets. It also works with the *RTP Packetizer* to encapsulate each NAL unit into the RTP payload format for H.264 video [10]. The *Demultiplexer* (DEMUX) then routes the important NAL units (SPS, PPS, SH, and prioritized I-frame data) through TCP and the rest of the NAL units through UDP. The *BP selection* module sets the BP parameter, which represents the percentage of the I-frame data to be prioritized and sent over TCP. In FDSP with Adaptive-BP, BP is adjusted dynamically based on the estimated available network bandwidth. Finally, *Dual Tunneling* is employed to keep both TCP and UDP sessions active during video streaming.

In the receiver, *Dual Tunneling* is employed to receive packets from both the TCP and UDP streams. The *Multiplexer* (MUX) then discards the late TCP packets and rearranges the TCP and UDP packets based on their RTP timestamps.

During FDSP streaming, the sender first segments the input video into 10 sec. *substreams*, as done in HTTP live Streaming (HLS) [11]. Then, all the TCP packets containing SPS, PPS, SH, and BP prioritized I-frame data for each substream are sent over TCP prior to sending UDP packets containing the slice data. Thus, the receiver must wait for its respective TCP data to arrive before playback. To avoid rebuffering caused by TCP packet delay, the transmission of UDP packets for the

 ISBN: 978-1-61208-548-7

current substream is overlapped with the transmission of TCP packets for the next substream (i.e., substream overlapping).

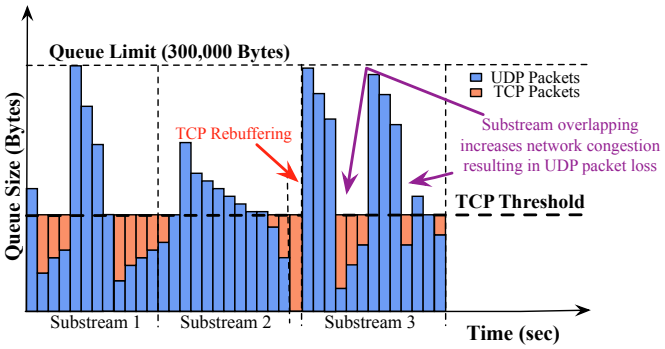
BP is only applied to packets containing I-frame data because they serve as reference frames and any loss in I-frame data leads to error propagation to the entire Group Of Picture (GOP) sequence. If the BP parameter is set to zero, then it defaults to basic FDSP, where SPS, PPS, and slice headers are the only packets that will be sent via TCP. If BP is 25% then a quarter of all I-frame packets would be sent via TCP. Although it is possible to select any distribution of the I-frame to be sent via TCP, a sequential order of I-frame packets are selected to be sent via TCP to achieve better QoE. Increasing BP results in increasing the number of TCP packets, thus increasing the probability of TCP rebuffering, but it reduces UDP packet loss and error propagation due to the proportional reduction in the number of UDP packets.

Since FDSP is a hybrid streaming technique that uses both TCP and UDP protocols, its performance is affected by both packet loss and rebuffering. The various factors that influence packet loss and rebuffering are the BP parameter, the substream length, and substream overlapping. The BP parameter is used to determine the percentage of I-frame packets that are to be sent through TCP. The BP parameter is computed based on TCP round-trip time and UDP packet loss rate, which in turn determines the percentage of TCP versus UDP packets to be sent for each substream. Adaptively adjusting the BP parameter for each substream helps further reduce UDP packet loss while keeping TCP rebuffering time and instances low. The substream length trades off between the likelyhood of rebuffering and the frequency of adaptive BP selection process. Since the BP and substream length have been analyzed in our previous work, this paper focuses on examining how different TCP thresholds affects FDSP video streaming.

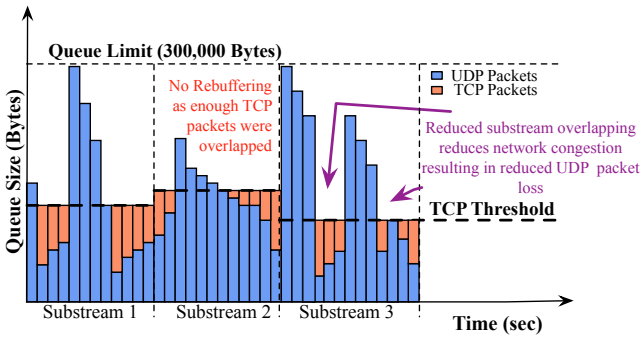
IV. Substream Overlapping and TCP Threshold

In FDSP, the important syntax elements for each substream are sent first via TCP, and then the rest of the data is sent via UDP. Therefore, the substream length and the BP parameter determine the amount of TCP data that needs to be sent prior to the playback of that substream. However, rebuffering occurs whenever TCP packets for a substream are not yet all available at the time of playback of that substream. To avoid rebuffering, the TCP packets of the next substream (i.e., substream $i+1$) are overlapped with the UDP packets of the current substream (i.e., substream i).

The important issue with substream overlapping is the decision on when to insert new TCP packets for the next substream into the outgoing IP queue of the sender. This is because inserting TCP packets for the next substream into the queue too soon would interfere with successful UDP packet transmission for the current substream. On the other hand, inserting TCP packets into the queue too late would result in rebuffering. Therefore, substream overlapping is initiated only when the number of packets in the network device's IP queue is below the *TCP Threshold*. Increasing the TCP threshold would result in increased UDP packet loss due to network saturation caused by flooding of TCP packets. On the other hand, decreasing the TCP threshold results in increased TCP rebuffering due to the reduction in the number of new TCP packets being inserted into the queue. The various factors that affect the TCP threshold are: (i) the number of UDP packets in the current substream; (ii) the number of TCP packets in the next substream; (iii) the average queue occupancy; and (iv) the average number of UDP packets per frame. The number of UDP packets for the current substream and the number of TCP packets for the next substream are used to calculate the estimated TCP rebuffering time and the estimated UDP PLR [4]. The average queue occupancy and the average number of UDP packets per frame are used to estimate the number of instances for substream overlapping.



(a) IP queue occupancy for static TCP threshold.



(b) IP queue occupancy for adaptive TCP threshold.

Figure 4. IP queue occupancy example for static vs. adaptive TCP threshold.

Figure 4a shows an example of the IP queue occupancy as a function of time for a static TCP threshold during FDSP streaming. In this example, the TCP threshold is set to 30% of the maximum queue size. For substream 1, the queue size decreases as UDP packets are streamed. When the queue size falls below the TCP threshold, the TCP packets for substream 2 are inserted into the queue. In substream 2, the average number of UDP packets per frame is higher than in substream 1 resulting in reduced opportunities to insert new TCP packets from substream 3, which in-turn increases the probability of TCP rebuffering (indicated by an extra time slot). In substream 3, both the queue occupancy and the average number of UDP packets per frame are high indicating a network congestion. In this scenario, inserting more TCP packets into the queue would exacerbate network congestion and result in UDP packet loss.

Figure 4a clearly shows that having a static TCP threshold is not always optimal for FDSP streaming. Figure 4b shows that adaptively adjusting the TCP threshold can reduce both UDP PLR and TCP rebuffering resulting in better QoE. In this example, the TCP threshold is also set to 30% by default for substream 1, and thus the queue behavior is identical to that of Figure 4a. In substream 2, the average number of UDP packets per frame is higher than substream 1, this reduces the number of TCP packet insertions resulting in

 ISBN: 978-1-61208-548-7

increased TCP rebuffering probability. However, since the queue occupancy for substream 2 is not very high, the TCP threshold is increased, which increases the number of TCP packets that can be inserted into the queue and in turn reduces the TCP rebuffering probability.

In substream 3, both the queue occupancy and the average number of UDP packets per frame are very high, indicating network congestion, which in-turn increases UDP packet loss. In Figure 4b, the TCP threshold is decreased for substream 3 to reduce the number of TCP packets inserted into the queue. This in-turn increases the number of UDP packets transmitted, thus reducing UDP PLR. Note that although the TCP threshold is decreased there is no change in the queue level for the first three time slots of substream 3 because there is no substream overlapping. During time slots 4-6, the number of TCP packets inserted into the queue is reduced compared to the case in Figure 4a. This reduction allows for more UDP packets to be transmitted. The increased UDP packet prioritization reduces the queue occupancy levels during time slots 7-9, which reduces the overall UDP packet loss rate for substream 3. This reduction in TCP threshold may increase the TCP rebuffering probability, but there are enough opportunities for TCP overlapping for substream 3 to avoid rebuffering.

A. Adaptive TCP Threshold

The Additive-Increase/Multiplicative-Decrease (AIMD) algorithm is well suited to adaptively adjust the TCP threshold because the TCP threshold is progressively increased, which in-turn increases the number of TCP packets inserted into the queue reducing the likelihood of rebuffering. On the other hand, AIMD prioritizes UDP packets by exponentially reducing the TCP threshold at the first sign of network congestion (based on estimated UDP packet loss). For example, FDSP streaming begins with a default TCP threshold of 30% for the first two substreams. From the third substream on, the TCP threshold is progressively incremented based on the estimated TCP rebuffering probability until UDP packet loss is estimated at which point the TCP threshold is reduced in half. Note that the estimated TCP rebuffering probability and the estimated UDP PLR are computed using TCP round trip time and queue statistics, respectively (for more details please refer to FDSP with Adaptive-BP [4]).

V. Experimental Setup

This section discusses the experimental setup for the analysis of TCP threshold parameters and its impact on both UDP packet loss and TCP rebuffering for FDSP-based video streaming. For our experiments, two full HD (1920×1080 @30fps, 4300 frames) clips from a high-motion (animation) video *Bunny*, and a low-motion (documentary) video *Bears* are used. These clips are encoded using the x264 encoder with an average bit rate of 4 Mbps and four slices per frame.

Our simulation environment is *Open Evaluation Framework For Multimedia Over Networks* (OEFMON) [12], which is composed of a multimedia framework *DirectShow*, and a network simulator *QualNet* 7.3 [13]. OEFMON allows a raw video to be encoded and redirected to a simulated network to gather statistics on the received video. Within OEFMON, an 802.11g ad-hoc network with a bandwidth of 54 Mbps is setup. Note that the version of the Qualnet simulator used for our study only supports the IEEE 802.11g standard. However, the simulation study can easily be adapted to 802.11n by having more background traffic to saturate the network. The network scenario used is an 8-node configuration shown in Figure 5. 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 primary test video is being streamed between nodes 1 and 2, while the remaining three node pairs produces three streams of constant bit rate (CBR) background traffic of 50 Mbps to fully saturate the network.

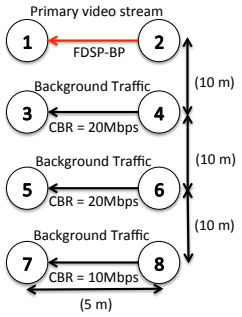


Figure 5. Simulated network scenario.

For the TCP threshold analysis, 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 [14], 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, the effects of UDP PLR and TCP rebuffering are effectively isolated via their corresponding BP values in order to study how TCP threshold changes affect FDSP streams. For each BP value, 20 different TCP threshold values, ranging from 5% to 100%, are evaluated.

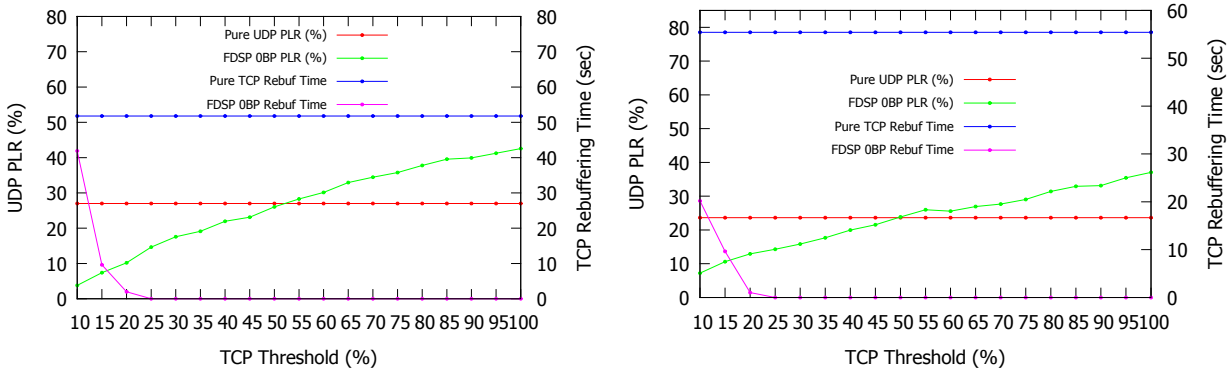
VI. Results

A. Impact of TCP threshold changes on UDP PLR and TCP Rebuffering

Figures 6 and 7 show the effects of the TCP threshold changes on both test videos in a fully congested network with BP of 0% and 100%, respectively. These figures also include the results for pure UDP and pure TCP as a comparison. In a fully congested network scenario, the total TCP rebuffering time incurred by both test videos that are streamed using FDSP with 0% BP decreases as the TCP threshold increases. Conversely, the UDP PLR incurred by both test videos increases as the TCP threshold increases. For example, in the *Bears* video (Figure 6a), TCP thresholds of 10%, 15%, and 20% incur UDP PLRs of 3.8%, 7.4%, and 10.2% and TCP rebuffering times of 43 seconds, 9.8 second, and 1.98 seconds, respectively. Similarly, in the *Bunny* video (Figure 6b), TCP thresholds of 10%, 15%, and 20% incur UDP PLRs of 7.2%, 10.62%, and 12.93% and TCP rebuffering times of 20.21 seconds, 9.64 seconds, and 1.05 seconds, respectively.

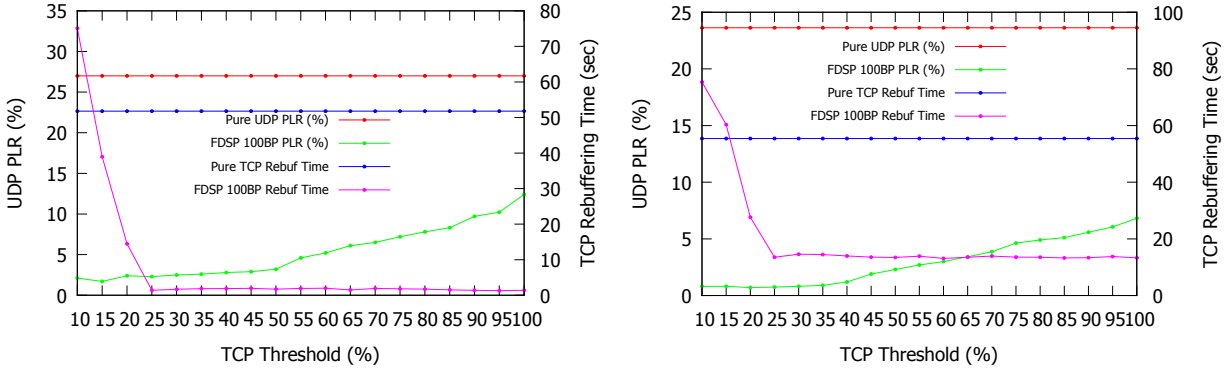
In comparison to FDSP with 0% BP streaming, UDP PLR and TCP rebuffering incurred by FDSP with 100% BP in a fully congested network scenario are much more pronounced. For example, in the *Bears* video (Figure 7a), TCP thresholds of 10%, 15%, and 20% incur UDP PLRs of 2.1%, 1.7%, and 2.4% and TCP rebuffering times of 75.1 seconds, 38.95 seconds, and 14.47 seconds, respectively. Similarly, in the *Bunny* video

 ISBN: 978-1-61208-548-7



(a) *Bears* video. (b) *Bunny* video.

Figure 6. Impact of TCP threshold changes on UDP packet loss and TCP rebuffering in a fully congested network scenario with BP of 0%.



(a) *Bears* video. (b) *Bunny* video.

Figure 7. Impact of TCP threshold changes on UDP packet loss and TCP rebuffering in a fully congested network scenario with BP of 100%.

(Figure 7b), TCP thresholds of 10%, 15%, and 20% incur UDP PLRs of 0.8%, 0.81%, and 0.74% and TCP rebuffering times of 75.38 seconds, 60.34 seconds, and 27.63 seconds, respectively. These results show that having a large TCP threshold results in greater opportunities to insert new TCP packets into the IP queue reducing TCP rebuffering time. This, in turn, will cause UDP PLR to increase due to the increase in UDP packet delay resulting in late packets. On the other hand, a smaller TCP threshold results in fewer opportunities to insert TCP packets into the IP queue. This means that fewer TCP packets are sent through substream overlapping and, instead they are buffered in between substreams, thus increasing the total TCP rebuffering time.

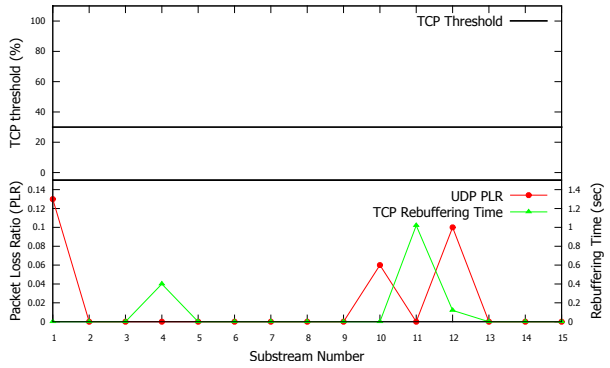
The ideal TCP threshold region is the one that minimizes both UDP PLR and TCP rebuffering. For the *Bears* video using FDSP with 0% BP (Figure 6a), the ideal TCP threshold region lies between 15% to 30%. For the *Bunny* video using FDSP with 0% BP (Figure 6b), the ideal TCP threshold region lies between 15% to 25%. Similarly, for the *Bears* video using FDSP with 100% BP (Figure 7a), the ideal TCP threshold region lies between 25 to 50%. For the *Bunny* video with FDSP 100% BP (Figure 7b), the ideal TCP threshold region lies between 25 to 55%. The optimal threshold range for both videos increases as BP increases to accommodate the increase in the number of TCP packets. These results show that the ideal TCP threshold is not constant for all types of videos. Furthermore, the ideal TCP threshold region is affected by the changes in the BP parameter. Hence, the TCP threshold has to be adaptively adjusted for each substream to optimize the substream overlapping and improve FDSP's performance.

B. *Performance of Adaptive TCP Threshold*

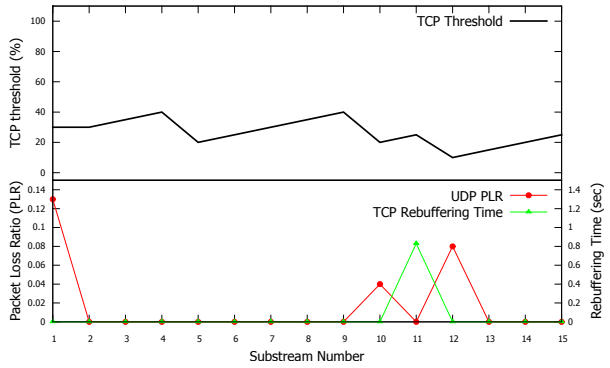
Figures 8a and 8b show UDP PLR and TCP rebuffering time for the *Bunny* video streamed based on FDSP with Adaptive-BP using static and adaptive TCP threshold techniques, respectively. Note that the figures are also stacked with plots of the TCP threshold values. These results show that adaptively adjusting the TCP threshold for each substream reduces both UDP PLR and TCP rebuffering time as compared to having a static TCP threshold. For example, using the adaptive TCP threshold scheme incurs only one instance of rebuffering that lasts for 0.83 seconds as compared to the static TCP threshold scheme, which incurs three instances of rebuffering with 0.4 seconds, 1.02 seconds, and 0.12 seconds of rebuffering times. Similarly, the adaptive TCP threshold scheme incurs three instances of UDP packet loss with PLRs of 0.13, 0.04, and 0.08, where as the static TCP threshold scheme also incurs three instances of UDP packet loss but with slightly higher PLRs of 0.13, 0.06 and 0.1. Thus, adaptively adjusting the TCP threshold reduces the TCP rebuffering incurred by 50% and also slightly reduces UDP PLR for the *Bunny* video.

Figures 9a and 9b compare the visual quality of the static and adaptive TCP threshold schemes for frames 2918 and 3391, respectively. These figures clearly show that the adaptive TCP threshold scheme achieves better visual quality than the static TCP threshold scheme by reducing UDP PLR as well as TCP rebuffering time. For frame 2918, the adaptive TCP

 ISBN: 978-1-61208-548-7



(a) Static TCP threshold results.



(b) Adaptive TCP threshold results.

Figure 8. Comparison of static vs. adaptive TCP threshold performance of *Bunny* video with Adaptive-BP.

threshold scheme incurs no packet loss resulting in perfect frame quality. For frame 3391, adaptively adjusting the TCP threshold results in slightly lower packet loss as compared to using a static TCP threshold value even though the difference in visual quality is marginal.

These results clearly show that the adaptive TCP threshold scheme reduces both UDP PLR and TCP rebuffering time as compared to the static TCP threshold scheme resulting in better end user QoE.

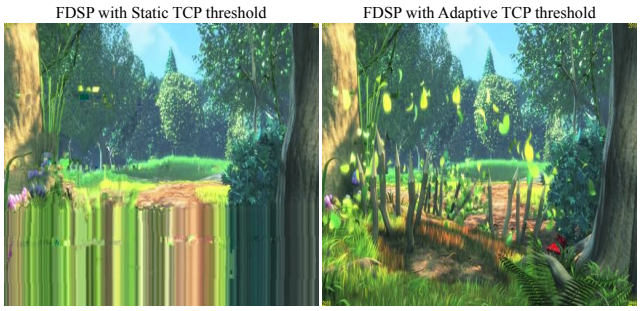
VII. Conclusion and Future Work

This paper studied the effects that different TCP threshold values have on video streaming in the context of Flexible Dual-TCP/UDP Streaming Protocol (FDSP). Our analysis showed that TCP threshold has a direct effect on both TCP rebuffering and UDP PLR. Our results showed that adaptively adjusting the TCP threshold using an AIMD algorithm reduces both packet loss ratio and rebuffering time, and leads to a better overall video streaming experience. As future work, we plan to study the impact of UDP packet loss and TCP rebuffering on end user Quality of Experience for FDSP streaming.

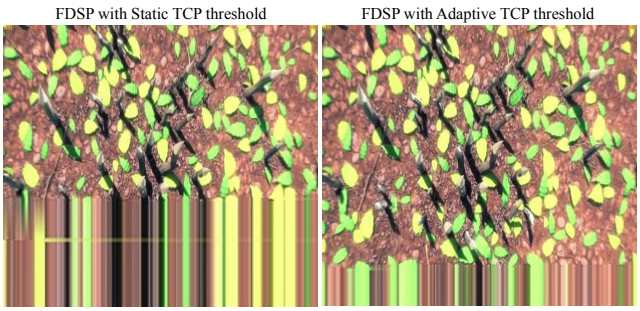
References

[1] C. Yoon, T. Um, and H. Lee, "Classification of n-screen services and its standardization," in 14th International Conference on Advanced Communication Technology (ICACT), 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 '13). New York, NY, USA: ACM, 2013, pp. 34:1–34:9.



(a) Video quality comparison of Frame 2918.



(b) Video quality comparison of Frame 3391.

Figure 9. Visual quality comparison of static vs. adaptive TCP threshold for frames 2918 and 3391.

[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 2015 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] S. Floyd and V. Jacobson, "Random early detection gateways for congestion avoidance," IEEE/ACM Transactions on Networking, vol. 1, no. 4, Aug 1993, pp. 397–413.

[6] W.-C. Feng, K. G. Shin, D. D. Kandlur, and D. Saha, "The blue active queue management algorithms," IEEE/ACM Transactions on Networking, vol. 10, no. 4, Aug 2002, pp. 513–528.

[7] K. Xu, M.Gerla, L. Qi, and Y. Shu, "Enhancing tcp fairness in ad hoc wireless networks using neighborhood red," in Proceedings of MOBICOM, San Diego, CA USA, sept 2003, pp. 16–28.

[8] J. Chen and V. C. M. Leung, "Applying active queue management to link layer buffers for real-time traffic over third generation wireless networks," in Wireless Communications and Networking, 2003. WCNC 2003. 2003 IEEE, vol. 3, March 2003, pp. 1657–1662 vol.3.

[9] M.-L. Shy, S.-C. Chen, and C. Ranasingha, "Router active queue management for both multimedia and best-effort traffic flows," in Multimedia and Expo, 2004. ICME '04. 2004 IEEE International Conference on, vol. 1, June 2004, pp. 451–454 Vol.1.

[10] 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.

[11] R. Pantos and W. May, "HTTP Live Streaming," Apr. 2014, iETF Draft, URL: https://developer.apple.com/streaming/ [Accessed: 2017-03-03].

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

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

[14] 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.

 ISBN: 978-1-61208-548-7