Experimental Study of Low-Latency HD VoD Streaming Flexible Dual TCP-UDP Streaming Protocol

Kevin Gatimu, Arul Dhamodaran, Taylor Johnson and Ben Lee

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

***Abstract*—The Flexible Dual TCP-UDP Protocol (FDSP) combines the reliability of TCP with the low latency characteristics of UDP. FDSP delivers the more critical parts of the video data via TCP and the rest via UDP. Bitstream Prioritization (BP) is a sliding scale that is used to determine the amount of TCP data that is to be sent. BP can be adjusted according to the level of network congestion. FDSP-based streaming achieves lower rebuffering time and less rebuffering instances than TCP-based streaming as well lower packet loss than UDP-based streaming. Our implementation and experiments on a real testbed shows that FDSP with BP delivers high quality, low-latency video, which is especially suitable for live video and subscription-based video.**

***Index Terms*—Low latency; HD Video Streaming; Hybrid Protocol; FDSP.**

I. Introduction

Global Internet traffic is projected to increase nearly three-fold until 2021, with video accounting for 82% of the total traffic [1]. Currently, consumer video is dominated by High Definition (HD), but higher resolutions such as 4K are gaining mainstream popularity [2]. Furthermore, there is an increasing number of video-capable devices and platforms being added globally everyday. For instance, the current 2 billion LTE subscribers are expected to double by 2021 [3]. Together, these factors will continue to increase global network congestion and pose even greater challenges to seamlessly delivering video at HD resolution and beyond.

This situation is further exacerbated by the unicast delivery model in major Video on Demand (VoD) services such as Netflix, Hulu, and Amazon Video, where each client requests video directly from a server. Therefore, as more clients connect to the server, the bandwidth requirements grow rapidly. VoD content providers have mitigated increased bandwidth demands by decentralizing their infrastructure through Content Delivery Networks (CDNs), which brings proxy servers closer to the end-user.

Another major development in managing VoD network resources is HTTP Adaptive Streaming (HAS). In HAS, the client requests video from a selection of multiple quality versions based on its perceived network conditions. Several HAS implementations exist, including proprietary ones such as Microsoft Smooth Streaming (MSS) [4], Adobe HTTP Dynamic Streaming [5], Apple's HTTP Live Streaming (HLS) [6], and the open-source standard, Dynamic Adaptive Streaming over HTTP (DASH) [7].

However, even the combination of HAS and CDNs is challenged by extremely large audiences, resulting in high bandwidth requirements for Internet video content providers. This is especially the case for live video streaming for events such as sports (e.g., the Olympics and the World Cup) and presidential debates. Furthermore, HAS suffers from high latency – often 20 seconds or more [8]. This is because two or more substreams, typically 10 seconds each, need to be buffered prior to playout. Such initial startup delay is acceptable for pre-recorded content (e.g., movies) as this maximizes the client's video quality with reduced rebuffering. However, the latency for live events needs to be minimized. Low latency is also required for subscription-based live video services such as Internet Protocol television (IPTV). When a client switches between different channels of streaming video, the transition needs to be as close as possible to traditional broadcast television with hardly any noticeable delay.

The Transmission Control Protocol (TCP) is the transport layer protocol used in HAS. When outstanding packets are acknowledged by the receiver, TCP additively increases the transmission rate of the sender by a constant amount. On the other hand, when acknowledgments are lost due to congestion, the sender retransmits the lost packets and halves the transmission rate. This is detrimental towards meeting playout deadlines for achieving low-latency video streaming. The User Datagram Protocol (UDP) is better suited for low-latency applications compared to TCP. As a result, there have been hybridization efforts at the transport layer in order to combine the reliability of TCP with the low latency of UDP pioneered by *Reliable UDP* [9] and culminating in the more advanced *Quick UDP Internet Connections* (QUIC) [10]. However, QUIC has been shown to have higher protocol overhead than TCP at low bitrates [11]. UDP has also been useful from an infrastructural point-of-view by supplementing CDNs with UDP-based peer-to-peer (P2P) networks [12], [13].

Based on the aforementioned discussion, the objective of this paper is to show that low-latency VoD streaming can be achieved using a hybrid streaming protocol called *Flexible Dual Streaming Protocol* (FDSP). Our previous work showed that FDSP is suitable for improving direct device-to-device streaming using simulation studies [14]–[16]. In this paper, FDSP is tailored for a physical testbed with network emulation for a VoD streaming environment. Our findings show that FDSP-based streaming achieves lower latency than pure-TCP-based streaming while having less packet loss than pure-UDP-based streaming.

II. Related Work

HAS is the most popular streaming mechanism for delivering Internet video today. For this reason, there has been research and development in trying to reduce the latency that is caused by video segmentation. A client maintains a video buffer of two or more segments of typically 10 seconds each [6], [17], which results in latency of 20 seconds or more. Reducing the segment size to just a few seconds can reduce the size of a client's playout buffer, which in turn reduces latency. However, this increases the total number of segments and, therefore, the number of HTTP requests that the client sends to the server in order to retrieve the video segments. These requests use precious bandwidth at a rate of one round-trip time (RTT) per video segment. For instance, a client that requests 2-second video segments on a network path with an RTT delay of 300 ms will experience 300 ms of additional delay every 2 seconds. In [18], Swaminathan *et al.* use HTTP chunked encoding to disrupt this correlation between live latency and segment duration by using partial HTTP responses. However, the persistent connections that are needed for chunked encoding transfer are prone to timeout issues and security concerns such as injection attacks and denial-of-service attacks [19]. Alternatively, HTTP/2 provides server push mechanisms such that the client receives multiple video segments per request [20]–[22]. However, HTTP/2 is not as widely available as legacy HTTP. HTTP/2 only has 15% worldwide deployment and at a current growth rate of 5% additional coverage every year, it has a long way to go before becoming a widely recognized standard [23].

Other improvements in reducing video latency include modifications to the transport layer. For instance, Chakareski *et al.* used multiple TCP connections in conjunction with Scalable Video Coding (SVC) [24]. More important packets were transmitted via better quality TCP connections and were, therefore, less prone to retransmissions. While this method addresses delay within the transport layer, there is still significant delay in the application layer due to the typical video segment sizes in HAS. On the other hand, Houze *et al.* proposed a multipath TCP streaming scheme based on the application layer, where larger video frames were subdivided based on media container formats [25]. They were then transmitted across two concurrent TCP connections and reassembled by the client. However, this method uses HTTP chunked encoding.

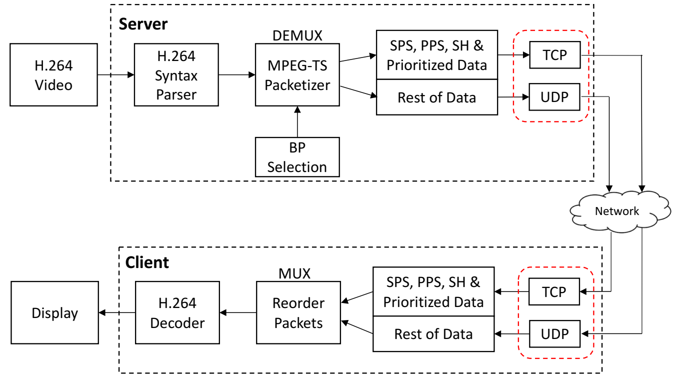


Fig. 1: Flexible Dual-UDP/TCP Streaming Protocol (FDSP) Architecture [14] augmented with modified MUX and DEMUX modules for FDSP-BP.

Peer-to-peer (P2P) networks have been used to supplement CDNs and help content providers save on deployment and maintenance costs [26]. This also reduces HTTP requests made to CDN servers thus lowering the latency for live streaming [27], [28]. In fact, CDN caching increases delay by 15-30 seconds [29]. CDN-P2P architectures have been commercialized for some time now by global CDN companies such as ChinaCache [12] and Akamai [13]. These hybrid architectures primarily rely on CDNs for HTTP-based retrieval of initial or critical video segments while using P2P networks for bandwidth relief or to retrieving future segments. Even though the P2P networks are UDP-based, standardized NAT/firewall traversal for UDP-based transmission is gaining traction primarily through WebRTC [30], which is a collection of protocols and browser APIs.

This paper shows that FDSP-based streaming achieves much lower latency compared to HTTP-based streaming at comparable video quality levels. Our study also shows that FDSP transmission results in lower packet loss compared to UDP-based streaming, even in congested networks. Furthermore, FDSP is orthogonal to adaptive streaming and can thus be used as a transport protocol for today's segment-based video delivery systems.

III. FDSP Overview

This section provides an overview of FDSP, including its architectural features and video streaming using substreams. For more details, see [14], [15] and [16]. FDSP is a hybrid streaming protocol that combines the reliability of TCP with the low latency characteristics of UDP. Figure 1 shows the FDSP architecture consisting of a server and a client.

At the server, the *H.264 Syntax Parser* processes video data in order to detect critical H.264 video syntax elements (i.e., Sequence Parameter Set (SPS), Picture Parameter Set (PPS), and slice headers). The *MPEG-TS Packetizer* within the *Demultiplexer* (DEMUX) then encapsulates all the data according to the RTP MPEG-TS specification. The DEMUX then directs the packets containing critical data to a TCP socket and the rest to the UDP socket as *Dual Tunneling* keeps

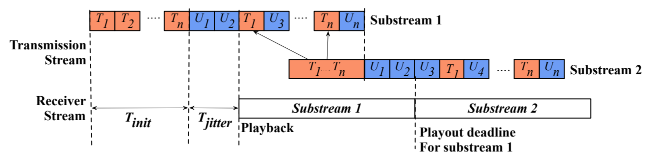


Fig. 2: Substream overlapping. Each packet is either UDP (U) or TCP (T), where the subscript represents the packet number within a substream.

Fig. 3: Experiment testbed.

both TCP and UDP sessions simultaneously active during video streaming. The *BP Selection* module sets the Bitstream Prioritization (BP) parameter, which is a percentage of I-frame data that is to be sent via TCP in addition to the original critical data. At the receiver, the *Multiplexer* (MUX) sorts TCP and UDP packets based on their RTP timestamps. This reordering is essential for the *H.264 Decoder* to decode incoming data correctly.

When a stream is initiated, the FDSP server transmits the packets for the first 10-second substream. All the TCP packets for this substream must be received (i.e., buffered) before playback begins. This startup delay is low since only the TCP portion of the data is sent rather than the whole 10 seconds of video. In order to minimize rebuffering, the TCP packets for the next substream are sent at the same time as the UDP packets for the current substream through a process called *substream overlapping* as illustrated in Figure 2. Substream overlapping is repeated throughout the duration of the stream. However, when playback for a particular substream is complete and the TCP packets for the upcoming substream are not yet all available, the client has to wait thus causing a rebuffering instance. The playout deadline for all subsequent packets is then incremented by the rebuffering time.

IV. Experiment Setup

Our experimental testbed is shown in Figure 3, which consists of a client-server pair and a traffic controller. The client-server pair is running *VLC Media Player* [31] on Mac OS X. The following modifications were made to integrate FDSP with BP into VLC:

1) Simultaneous streaming via UDP and TCP protocols.
2) Parsing H.264 video data at the server and subdividing it into TCP-bound and UDP-bound elements.
3) Reordering TCP and UDP packets and reconstructing the H.264 bitstream at the client prior to decoding.

The traffic controller, running on CentOS, connects the server to the client via a network bridge across interfaces `eth2` and `eth3`, respectively. The Linux *traffic control* (tc) utility was then used to perform traffic control on the network

Parameters	Value(s)
Bridge interface	eth2, eth3
Delay (ms)	0, 25, 50, 75, 100, 125
Jitter (ms)	0, 5, 10, 15, 25
Loss	0.2%
Duplicate	0.2%
Corrupt	0.2%
Reorder	0.2%

TABLE I: Network emulation settings for traffic control (tc).

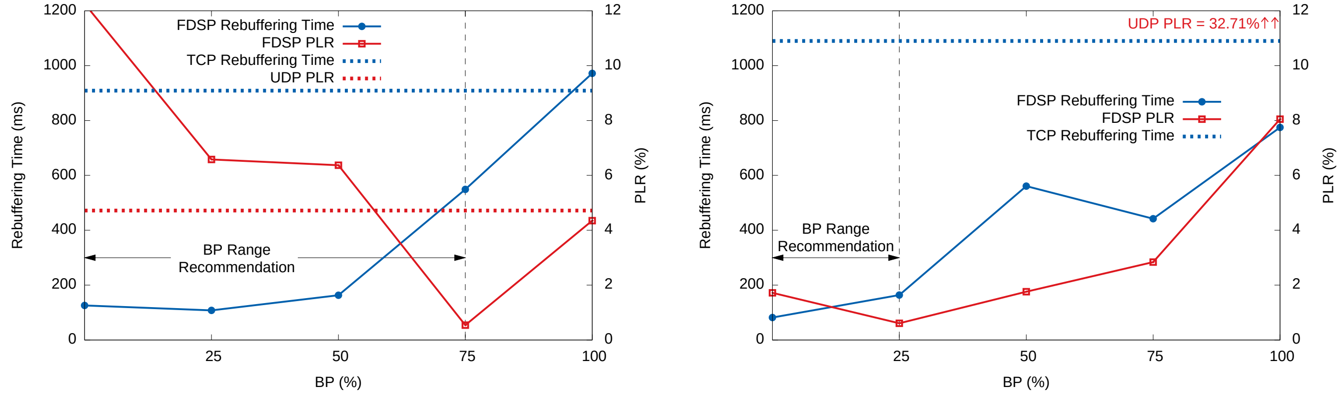
bridge. The *tc* configures the Linux kernel primarily through *queueing disciplines* (qdiscs). A *qdisc* is an interface between the kernel and a network interface, where packets are queued and released according to *tc* settings. For example, a loss setting drops packets from the *qdisc* according to a specified percentage, while a delay setting keeps the packets in the *qdisc* longer. Multiple settings can be used together. A summary of *tc* settings is shown in Table I.

The *tc* parameters chosen represent an array of Wide Area Network (WAN) scenarios, which would typically plague Internet video streaming performance. The *Delay* setting was primarily used to simulate different levels of real-world Internet congestion [32]. The core network RTT latency is about 30 ms within Europe, 45 ms within North America, and 90 ms for Trans-Atlantic routes [33]. However, the edge network introduces additional latency. Therefore, *Delay* ranging from 0 to 125 ms in increments of 25 ms were used for each of the two bridged interfaces (`eth2` and `eth3`), resulting in a total RTT delay range of 0 to 250 ms. The corresponding random *Jitter* value was set at 20% of the delay. The *Duplicate* setting simulates duplicate packets, e.g., due to TCP retransmissions. The *Loss* setting simulates packets randomly dropped by the network. The *Corrupt* setting introduces a random bit error in a specified percentage of the packets. Finally, the *Reorder* setting simulates multi-hop routing by further delaying a specified percentage of packets according to the delay and jitter settings.

The test videos used for streaming are two full HD (1920×1080 @30fps) 30-second clips from an animation video, *Bunny*, and a documentary video, *Nature*. These videos are encoded using x264 with an average bit rate of 4 Mbps and four slices per frame. They are then streamed from the server to the client using FDSP, TCP, and UDP. For each streaming protocol, the five different levels of network congestion are created via the network delay settings (i.e., 50 ms, 100 ms, 150 ms, 200 ms, and 250 ms). Furthermore, FDSP-based streaming is done for five different BP values (i.e., 0%, 25%, 50%, 75%, and 100%) per congestion level.

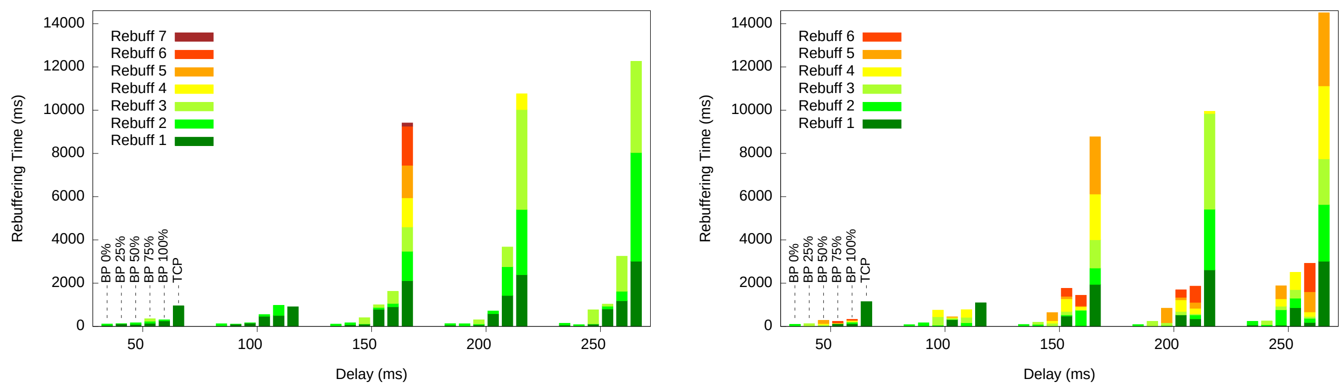
V. Results

This section discusses the results of our experiments. FDSP-based streaming generally outperforms TCP-based streaming in terms of both rebuffering time and number of rebuffering instances. FDSP also incurs lower PLR than UDP. Figure 4 shows a sample of the video streaming improvements of FDSP over either TCP or UDP at 100 ms delay. The other levels of network congestion show similar results. Overall, FDSP rebuffering time is significantly lower than TCP rebuffering



(a) *Nature* video (b) *Bunny* video

Fig. 4: Rebuffering time and PLR for FSDP, TCP and UDP at 100 ms delay.



(a) *Nature* video (b) *Bunny* video

Fig. 5: Rebuffering for different levels of network congestion for FDSP-based streaming at different values of BP and TCP-based streaming.

time. In addition, as BP increases within a recommended range, PLR decreases. The BP range recommendations are 0% to 75% for *Nature* and 0% to 25% for *Bunny*. Since the overall rebuffering of FDSP-based streaming is significantly lower than that of TCP-based streaming, BP range recommendation was based on minimizing PLR. The rest of this section discusses the two major improvements, i.e., lower rebuffering and lower PLR.

A. *FDSP Improvement over TCP in Rebuffering*

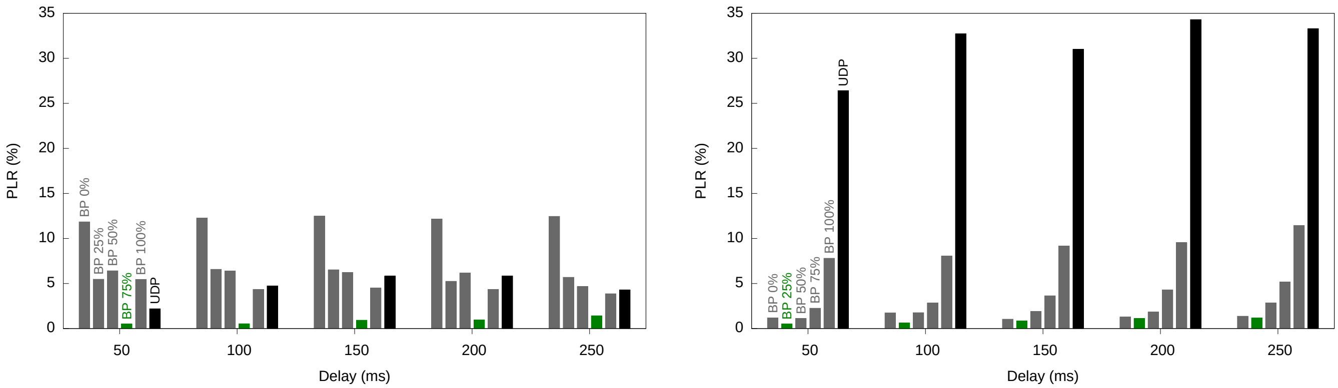
Reduction in both rebuffering time and instances is important towards improving the user's Quality of Experience (QoE). Figure 5 shows the total amount of rebuffering time and the number of rebuffering instances for the different levels of network congestion. For each congestion level, rebuffering is shown for FDSP with different values of BP as well for TCP. For instance, in *Nature* at 150 ms delay, FDSP rebuffering time ranges from 108 ms to 1,616 ms, compared to 9,410 ms in TCP. In addition, the number of rebuffering instances ranges from 2 to 3 for FDSP compared to 7 for TCP. Meanwhile, in *Bunny* at 150 ms delay, FDSP rebuffering time ranges from 92 ms to 1,441 ms with 1 to 6 instances, compared to 8,764 ms with 5 instances for TCP. Note that the first rebuffering instance (*Rebuff 1* in Figure 5) is the startup delay. As can be seen, FDSP exhibits lower startup delay than TCP at almost all BP levels.

While FDSP is significantly better than TCP in terms of rebuffering, it is important to note that rebuffering does increase with BP.

B. *FDSP Improvement over UDP in PLR*

FDSP-based streaming results in not only less rebuffering, but it also produces better video quality by reducing PLR. Figure 6 shows the effect of BP on PLR across different levels of network congestion for both *Nature* and *Bunny*. For each congestion level, PLR is shown for FDSP with different values of BP as well as for UDP. As BP increases, there is less PLR and thus better video quality. For *Nature*, the best BP value is 75% while for *Bunny* it is 25%. This implies that there is an optimal range of BP values based on the type of video.

As BP increases within the optimal range, more packets are sent via TCP rather than UDP. This protects them from network-induced losses. Since the bulk of PLR is due to lost UDP packets, the overall PLR decreases as BP increases. For example, in *Nature*, the PLR at 50 ms delay decreases from 9% to 0.32% as BP increases from 0% to 75%. Similarly,



(a) *Nature* video

(b) *Bunny* video

Fig. 6: PLR for different levels of network congestion for FDSP-based streaming at different values of BP and UDP-based streaming.

(a) UDP

(b) Basic FDSP (0% BP)

Fig. 7: Visual comparison between UDP-based streaming and FDSP-based streaming for *Bunny*.

in *Bunny*, the PLR decreases from 1.19% to 0.51% as BP increases from 0% to 25%. Figure 7 shows a sample of the visual improvement of FDSP-based streaming with 0% BP over pure-UDP streaming in *Bunny*. The frame in Figure 7b is intact while the frame in Figure 7a shows the effects of packet loss under UDP-based streaming. In such situations, the loss of just a slice header or the first few bytes of a slice renders the rest of the slice data useless to the decoder, thus resulting in error concealment as shown in slice 4 of Figure 7a. On the other hand, FDSP-based streaming, even with no BP, protects slice headers through TCP transmission thus producing better quality video frames as shown in Figure 7b.

If BP surpasses the optimal range and becomes too high, the network can become saturated with TCP packets. This is because when there is network congestion, more packets are delayed, reordered or lost. The TCP packets are then more prone to retransmissions so as to guarantee in-order, reliable delivery. Meanwhile, the IP queue is filled with staged TCP and UDP packets. As the IP queue fills up with TCP packets, additional UDP packets are dropped. This is the cause of most of the PLR when BP becomes too high. In addition, some packets (both UDP and TCP) arrive at the client too late, past the decoder's playout deadline, and are thus also considered lost.

The frequency of I-frames can be used to categorize the type of video and determine the optimal range of BP. For videos such as *Bunny*, where there are many scene changes, there is usually a corresponding higher number of I-frames. In fact, there are 37 I-frames in *Bunny* compared to just 5 in *Nature*. Since I-frames contain significantly more data than other frames, the probability of network saturation increases with the frequency of I-frames, which leads to high PLR. For instance, Figure 6 shows much higher PLR for UDP-based streaming in *Bunny* (26.4%∼33.3%) compared to *Nature* (2.2%∼4.3%). In such scenarios (*Bunny*), small BP values (0%∼25%) are effective towards reducing PLR while higher values (>25%) will saturate the network with TCP packets from I-frame data.

In comparison, videos exemplified by *Nature* have lower PLR to begin with for UDP-based streaming. This is because of less network saturation as a result of lower I-frame frequency. When such videos are streamed through FDSP, the introduction of TCP packets increases the likelihood of network saturation and UDP PLR. However, higher BP values (up to 75% in the case of *Nature*) can be applied to the point of lowering UDP PLR below that of UDP-based streaming.

VI. Conclusion and Future Work

This paper shows that the FDSP with BP is suitable for low-latency HD video streaming over the Internet while maintaining high video quality by combining the reliability of TCP with the low-latency characteristics of UDP. Our implementation and experiments on a real testbed consisting of a server and a client and an intermediate node for network emulation through the Linux traffic control utility showed that FDSP with BP results in significantly less rebuffering than TCP-based streaming and much lower PLR that UDP-based streaming.

As future work, BP will be dynamically adjusted with varying network conditions. A separate QoE study based on FDSP streaming is currently in progress. Its results will be used to determine when BP should be changed based on variation in PLR and rebuffering.

References

[1] "VNI Global Fixed and Mobile Internet Traffic Forecasts." [Online]. Available: http://www.cisco.com/c/en/us/solutions/service-provider/visual-networking-index-vni/index.html

[2] "4k Internet TV & Video to be Viewed by 1 in 10 US Residents," Aug. 2016. [Online]. Available: https://www.juniperresearch.com/press/press-releases/4k-internet-tv-video-content-to-be-viewed-by-1-i

[3] "Ericsson Mobility Report – Ericsson," Nov. 2016. [Online]. Available: https://www.ericsson.com/en/mobility-report

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

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

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

[7] "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

[8] "What YOU Need to Know About HLS: Pros and Cons," Jan. 2016. [Online]. Available: http://blog.red5pro.com/what-you-need-to-know-about-hls-pros-and-cons/

[9] T. Bova and T. Krivoruchka, "Reliable UDP Protocol." [Online]. Available: https://tools.ietf.org/html/draft-ietf-sigtran-reliable-udp-00

[10] A. Wilk, J. Iyengar, I. Swett, and R. Hamilton, "QUIC: A UDP-Based Secure and Reliable Transport for HTTP/2." [Online]. Available: https://tools.ietf.org/html/draft-hamilton-early-deployment-quic-00

[11] C. Timmerer and A. Bertoni, "Advanced Transport Options for the Dynamic Adaptive Streaming over HTTP," *arXiv preprint arXiv:1606.00264*, 2016. [Online]. Available: https://arxiv.org/abs/1606.00264

[12] X. Liu, H. Yin, and C. Lin, "A Novel and High-Quality Measurement Study of Commercial CDN-P2p Live Streaming," in *2009 WRI International Conference on Communications and Mobile Computing*, vol. 3, Jan. 2009, pp. 325–329.

[13] Z. Lu, Y. Wang, and Y. R. Yang, "An Analysis and Comparison of CDN-P2p-hybrid Content Delivery System and Model," *Journal of Communications*, vol. 7, no. 3, Mar. 2012. [Online]. Available: http://www.jocm.us/index.php?m=content&c=index&a=show&catid=39&id=90

[14] 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.

[15] 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.

[16] 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.

[17] 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

[18] V. Swaminathan and S. Wei, "Low latency live video streaming using HTTP chunked encoding," in *2011 IEEE 13th International Workshop on Multimedia Signal Processing*, Oct. 2011, pp. 1–6.

[19] G. Wilkins, S. Salsano, S. Loreto, and P. Saint-Andre, "Known Issues and Best Practices for the Use of Long Polling and Streaming in Bidirectional HTTP." [Online]. Available: https://tools.ietf.org/html/rfc6202#page-16

[20] S. Wei and V. Swaminathan, "Low Latency Live Video Streaming over HTTP 2.0," in *Proceedings of Network and Operating System Support on Digital Audio and Video Workshop*, ser. NOSSDAV '14. New York, NY, USA: ACM, 2014, pp. 37:37–37:42. [Online]. Available: http://doi.acm.org/10.1145/2578260.2578277

[21] W. Cherif, Y. Fablet, E. Nassor, J. Taquet, and Y. Fujimori, "DASH Fast Start Using HTTP/2," in *Proceedings of the 25th ACM Workshop on Network and Operating Systems Support for Digital Audio and Video*, ser. NOSSDAV '15. New York, NY, USA: ACM, 2015, pp. 25–30. [Online]. Available: http://doi.acm.org/10.1145/2736084.2736088

[22] R. Huysegems, J. van der Hooft, T. Bostoen, P. Rondao Alface, S. Petrangeli, T. Wauters, and F. De Turck, "HTTP/2-Based Methods to Improve the Live Experience of Adaptive Streaming," in *Proceedings of the 23rd ACM International Conference on Multimedia*, ser. MM '15. New York, NY, USA: ACM, 2015, pp. 541–550. [Online]. Available: http://doi.acm.org/10.1145/2733373.2806264

[23] A. Theedom. (2016) Tracking HTTP/2 Adoption: Stagnation - DZone Web Dev. [Online]. Available: https://dzone.com/articles/tracking-http2-adoption-stagnation

[24] J. Chakareski, R. Sasson, A. Eleftheriadis, and O. Shapiro, "System and method for low delay, interactive communication using multiple TCP connections and scalable coding," U.S. Patent US8 699 522 B2, Apr., 2014, u.S. Classification 370/474, 370/536, 375/240.05, 709/231; International Classification H04J3/24; Cooperative Classification H04L65/607, H04L47/32, H04L47/10, H04L47/193, H04L47/2416, H04L65/4015, H04L47/283, H04L65/80. [Online]. Available: http://www.google.com/patents/US8699522

[25] P. Houzé, E. Mory, G. Texier, and G. Simon, "Applicative-layer multipath for low-latency adaptive live streaming," in *2016 IEEE International Conference on Communications (ICC)*, May 2016, pp. 1–7.

[26] D. Xu, S. S. Kulkarni, C. Rosenberg, and H.-K. Chai, "Analysis of a CDN–P2p hybrid architecture for cost-effective streaming media distribution," *Multimedia Systems*, vol. 11, no. 4, pp. 383–399, Apr. 2006. [Online]. Available: https://link.springer.com/article/10.1007/s00530-006-0015-3

[27] S. M. Y. Seyyedi and B. Akbari, "Hybrid CDN-P2p architectures for live video streaming: Comparative study of connected and unconnected meshes," in *2011 International Symposium on Computer Networks and Distributed Systems (CNDS)*, Feb. 2011, pp. 175–180.

[28] T. T. T. Ha, J. Kim, and J. Nam, "Design and Deployment of Low-Delay Hybrid CDN–P2P Architecture for Live Video Streaming Over the Web," *Wireless Personal Communications*, vol. 94, no. 3, pp. 513–525, Jun. 2017. [Online]. Available: https://link.springer.com/article/10.1007/s11277-015-3144-1

[29] C. Michaels. (2017, February) HLS Latency Sucks, But Here's How to Fix It | Wowza. [Online]. Available: https://www.wowza.com/blog/hls-latency-sucks-but-heres-how-to-fix-it

[30] "WebRTC 1.0: Real-time Communication Between Browsers," 2017. [Online]. Available: https://www.w3.org/TR/webrtc/

[31] (2011). [Online]. Available: http://www.videolan.org/

[32] "Network Latency and Packet Loss Emulation @ Calomel.org." [Online]. Available: https://calomel.org/network_loss_emulation.html

[33] (2017, May) IP Latency Statistics. [Online]. Available: http://www.verizonenterprise.com/about/network/latency/