Performance Characterization of Streaming Video over Multipath TCP

Ryota Matsufuji, Dirceu Cavendish, Kazumi Kumazoe, Daiki Nobayashi, Takeshi Ikenaga, Yuji Oie
Department of Computer Science and Electronics
Kyushu Institute of Technology
Fukuoka, Japan
e-mail: {q349428r@mail, cavendish@ndrc, kuma@ndrc, nova@ecs, ike@ecs, oie@ndrc}.kyutech.ac.jp

***Abstract*—Video streaming has become the major source of Internet traffic nowadays. Considering that content delivery network providers have adopted Video over Hypertext Transfer Protocol/Transmission Control Protocol (HTTP/TCP) as the preferred protocol stack for video streaming, understanding TCP performance in transporting video streams has become paramount. Recently, multipath transport protocols have become available. In this paper, we evaluate the performance of Multipath TCP in conjunction with various TCP variants in transporting video streams over multiple paths. We utilize network performance measurers, as well as video quality metrics, to characterize the performance and interaction between network and application layers of video streams for various network scenarios. Overall, Cubic delivers best streaming performance over various path scenarios.**

***Keywords*—Video streaming; high speed networks; TCP congestion control; Multipath TCP; Packet retransmissions; Packet loss.**

I. INTRODUCTION

Transmission control protocol (TCP) is the dominant transport protocol of the Internet, providing reliable data transmission for the large majority of applications. For data applications, the perceived quality of experience is the total transport time of a given file. For real time (streaming) applications, the perceived quality of experience involves not only the total transport time, but also the amount of data discarded at the client due to excessive transport delays, as well as rendering stalls due to the lack of timely data. Transport delays and data starvation depend on how TCP handles flow control and packet retransmissions. Therefore, video streaming user experience depends heavily on TCP performance.

TCP protocol interacts with video application in non trivial ways. Widely used video codecs, such as H-264, use compression algorithms that result in variable bit rates along the playout time. In addition, TCP has to cope with variable network bandwidth along the transmission path. Network bandwidth variability is particularly wide over paths with wireless access links of today, where multiple transmission modes are used to maintain steady packet error rate under varying interference conditions. As the video playout rate and network bandwidth are independent, it is the task of the transport protocol to provide a timely delivery of video data so as to support a smooth playout experience.

Recently, a new transport paradigm has been proposed, which uses multiple paths to deliver data across the Internet. The idea is to take advantage of multiple IP interfaces and radios in modern devices to provide a robust transport protocol. Although multiple path transport brings the advantage of augmented aggregated bandwidth at the application layer, the main benefit to users might be the ability to maintain a transport level session even when a specific radio link coverage gets compromised. For instance, a video streaming session initiated at home on a WiFi link may be sustained long after the device is out of the access point coverage, if a cellular link is available. Another compelling use case is with docking stations of today, where a docked laptop loses internet connectivity every time it is undocked, even though a WiFi interface or even a cellular interface may be available. With multipath transport standards being developed, it is likely that data transport over multiple paths become mainstream in the near future.

In the last decade, many TCP variants have been proposed, mainly motivated by data transfer performance reasons. Most of the proposals deal with congestion window size adjustment mechanism, which is called congestion avoidance phase of TCP, since congestion window size controls the amount of data injected into the network at a given time. In previous works, we have studied TCP performance of most popular TCP variants - Reno [1], Cubic (Linux) [12], Compound (Windows) [13] - as well as our proposed TCP variants: Capacity and Congestion Probing (CCP) [2], and Capacity Congestion Plus Derivative (CCPD) [3], in transmitting data [4] and video streaming [5] over wireless path conditions. Our proposed CCP and CCPD TCP variants utilize delay based congestion control mechanism, and hence are resistant to random packet losses common in wireless links. We have also proposed TCP congestion avoidance enhancements to improve performance of video streaming [6] [7] on single paths. In this paper, we study the transport of video streams over multiple transport paths using widely deployed TCP variants.

The material is organized as follows. Related work discussion is provided on Section II. Section III describes video streaming over TCP system. Section IV introduces the TCP variants addressed in this paper, as well as Multipath TCP used to support multipath transport. Section V addresses multiple path video delivery performance evaluation for each TCP variant. Section VI addresses directions we are pursuing as follow up to this work.

II. RELATED WORK

Although multipath transport studies abound in the literature, only recently has streaming video performance over multiple paths been addressed. Park et. al. [10] seek to improve video streaming performance by streaming over multiple paths, as well as adapting video transmission rates to the

 ISBN: 978-1-61208-516-6

network bandwidth available. Such approach, best suited to distributed content delivery systems, requires coordination between multiple distribution sites. In contrast, we seek to understand network transport session carrying a video session by characterizing underneath TCP variants, independently of the video encoder.

Wu et. al. [14] advocate the use of a Forward Error Correction (FEC) coding to remedy artifacts on video streaming due to packet retransmissions on stringent delay constraint scenarios. Their framework seeks to improve video quality by formulating a combined FEC and path rate allocation optimization problem which takes into account paths packet loss, latency, as well as available bandwidth. Video codec, as well as MPTCP resource allocation, are affected, although TCP variants' impact on performance is not investigated, as in this paper.

Corbillon et al. [8] propose a cross layer scheduler which prioritizes video frames with best chance of being played out on time. Hence, late frames are discarded at the source, whereas frames with tight deadlines are given delivery priority. The approach requires coupling between application and MPTCP transport layers. In contrast, we evaluate video streaming performance of video/transport stacks that operate independently, focusing instead on performance differences due to popular TCP variants.

A distinct aspect of our current work is that we analyze the performance of video streaming over multipath TCP using widespread TCP variants, evaluating them on real client and server network stacks widely deployed for video streaming via VLC open source video client and standard HTTP server.

III. Video Streaming over TCP

Video streaming over HTTP/TCP involves an HTTP server side, where video files are made available for streaming upon HTTP requests, and a video client, which places HTTP requests to the server over the Internet, for video streaming. Fig. 1 illustrates video streaming components.

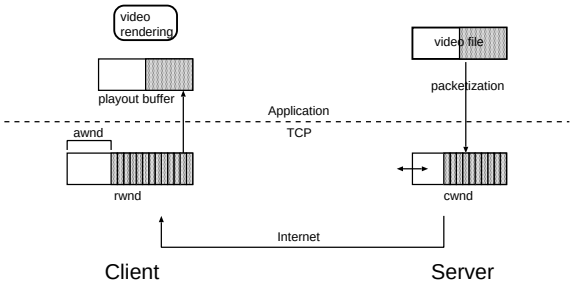


Figure 1: Video Streaming over TCP

An HTTP server stores encoded video files, available upon HTTP requests. Once a request is placed, a TCP sender is instantiated to transmit packetized data to the client machine. At TCP transport layer, a congestion window is used for flow controlling the amount of data injected into the network. The size of the congestion window, $cwnd$, is adjusted dynamically, according to the level of congestion in the network, as well as the space available for data storage, $awnd$, at the TCP client receiver buffer. Congestion window space is freed only when data packets are acknowledged by the receiver, so that lost packets are retransmitted by the TCP layer. At the client side, in addition to acknowledging arriving packets, TCP receiver sends back its current available space $awnd$, so that at the sender side, $cwnd \leq awnd$ at all times. At the client application layer, a video player extracts data from a playout buffer, filled with packets delivered by TCP receiver from its buffer. The playout buffer is used to smooth out variable data arrival rate.

A. Interaction between Video streaming and TCP

At the server side, HTTP server retrieves data into the TCP sender buffer according with $cwnd$ size. Hence, the injection rate of video data into the TCP buffer is different than the video variable encoding rate. In addition, TCP throughput performance is affected by the round trip time of the TCP session. This is a direct consequence of the congestion window mechanism of TCP, where only up to a $cwnd$ worth of bytes can be delivered without acknowledgements. Hence, for a fixed $cwnd$ size, from the sending of the first packet until the first acknowledgement arrives, a TCP session throughput is capped at $cwnd/rtt$. For each TCP congestion avoidance scheme, the size of the congestion window is computed by a specific algorithm at time of packet acknowledgement reception by the TCP source. However, for all schemes, the size of the congestion window is capped by the available TCP receiver space $awnd$ sent back from the TCP client.

At the client side, the video data is retrieved by the video player into a playout buffer, and delivered to the video renderer. Playout buffer may underflow, if TCP receiver window empties out. On the other hand, playout buffer overflow does not occur, since the player will not pull more data into the playout buffer than it can handle.

In summary, video data packets are injected into the network only if space is available at the TCP congestion window. Arriving packets at the client are stored at the TCP receiver buffer, and extracted by the video playout client at the video nominal playout rate.

IV. Anatomy of transmission control protocol

TCP protocols fall into two categories, delay and loss based. Advanced loss based TCP protocols use packet loss as primary congestion indication signal, performing window regulation as $cwnd_k = f(cwnd_{k-1})$, being ack reception paced. Most f functions follow an Additive Increase Multiplicative Decrease strategy, with various increase and decrease parameters. TCP NewReno [1] and Cubic [12] are examples of additive increase multiplicative decrease (AIMD) strategies. Delay based TCP protocols, on the other hand, use queue delay information as the congestion indication signal, increasing/decreasing the window if the delay is small/large, respectively. Compound [13], CCP [2] and CCPD [3] are examples of delay based protocols.

Most TCP variants follow TCP Reno phase framework: slow start, congestion avoidance, fast retransmit, and fast recovery.

 ISBN: 978-1-61208-516-6

- **Slow Start(SS):** This is the initial phase of a TCP session. In this phase, for each acknowledgement received, two more packets are allowed into the network. Hence, congestion window $cwnd$ is roughly doubled at each round trip time. Notice that $cwnd$ size can only increase in this phase. So, there is no flow control of the traffic into the network. This phase ends when $cwnd$ size reaches a large value, dictated by $ssthresh$ parameter, or when the first packet loss is detected, whichever comes first. All widely used TCP variants use slow start except Cubic [12].
- **Congestion Avoidance(CA):** This phase is entered when the TCP sender detects a packet loss, or the $cwnd$ size reaches the target upper size $ssthresh$ (slow start threshold). The sender controls the $cwnd$ size to avoid path congestion. Each TCP variant has a different method of $cwnd$ size adjustment.
- **Fast Retransmit and fast recovery(FR):** The purpose of this phase is to freeze all $cwnd$ size adjustments in order to take care of retransmissions of lost packets.

For TCP variants widely used today, congestion avoidance phase is sharply different. In what follows, we briefly introduce these TCP variants' congestion avoidance phase.

A. Multipath TCP

Multipath TCP (MPTCP) is a transport layer protocol, currently being evaluated by IETF, which makes possible data transport over multiple TCP sessions [9]. The key idea is to make multipath transport transparent to upper layers, hence presenting a single TCP socket to applications. Under the hood, MPTCP works with TCP variants which are unaware of the multipath nature of the overall transport session. To accomplish that, MPTCP supports a packet scheduler that extracts packets from the MPTCP socket exposed to applications, and inject them into TCP sockets belonging to a "sub-flow" defined by a single path TCP session. MPTCP transport architecture is represented in Fig. 2.

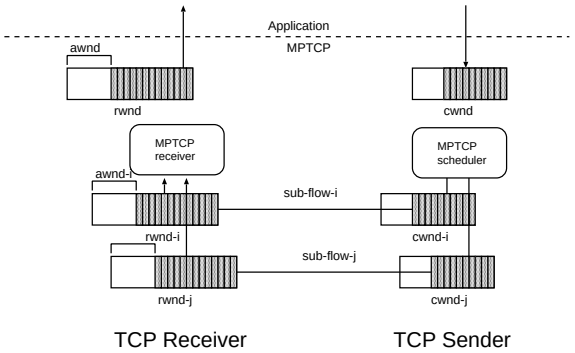


Figure 2: MPTCP Architecture

MPTCP packet scheduler works in two different configuration modes: uncoupled, and coupled. In uncoupled mode, each sub-flow congestion window $cwnd$ is adjusted independently. In coupled mode, MPTCP couples the congestion control of the sub-flows, by adjusting the congestion window $cwnd_k$ of a sub-flow k according with parameters of all sub-flows. Although there are several coupled mechanisms, we focus on Linked Increase Algorithm (LIA) [11]. In both cases, MPTCP scheduler selects a sub-flow for packet injection according to the shortest average packet round trip time (rtt) among all sub-flows with large enough $cwnd$ to allow packet injection.

MPTCP supports the advertisement of IP interfaces available between two endpoints via specific TCP option signalling. Both endpoints require MPTCP to be running for the establishment of multiple transport paths. In addition, IP interfaces may be of diverse nature: WiFi, cellular, etc.

B. Cubic TCP Congestion Avoidance

TCP Cubic is a loss based TCP that has achieved widespread usage as the default TCP of the Linux operating system. During congestion avoidance, its congestion window adjustment scheme is:

$$\begin{aligned} AckRec: \quad cwnd_{k+1} &= C(t-K)^3 + Wmax \\ K &= (Wmax\frac{\beta}{C})^{1/3} \\ PktLoss: \quad cwnd_{k+1} &= \beta cwnd_k \\ Wmax &= cwnd_k \end{aligned} \tag{1}$$

where C is a scaling factor, Wmax is the cwnd value at time of packet loss detection, and t is the elapsed time since the last packet loss detection (cwnd reduction). The rational for these equations is simple. Cubic remembers the cwnd value at time of packet loss detection - Wmax, when a sharp cwnd reduction is enacted, tuned by parameter β. After that, cwnd is increased according to a cubic function, whose speed of increase is dictated by two factors: i) how long it has been since the previous packet loss detection, the longer the faster ramp up; ii) how large the cwnd size was at time of packet loss detection, the smaller the faster ramp up. The shape of Cubic cwnd dynamics is typically distinctive, clearly showing its cubic nature. Notice that upon random loss, Cubic strives to return cwnd to the value it had prior to loss detection quickly, for small cwnd sizes.

Cubic fast release fast recovery of bandwidth makes it one of the most aggressive TCP variants. Being very responsive, it quickly adapts to variations in network available bandwidth. However, because it relies on packet loss detection for $cwnd$ adjustments, random packet losses in wireless links may still impair Cubic's performance.

C. Compound TCP Congestion Avoidance

Compound TCP is the TCP of choice for most deployed Wintel machines. It implements a hybrid loss/delay based congestion avoidance scheme, by adding a delay congestion window dwnd to the congestion window of NewReno [13]. Compound TCP cwnd adjustment is as per (2):

$$\begin{aligned} AckRec: \quad cwnd_{k+1} &= cwnd_k + \frac{1}{cwnd_k + dwnd_k} \\ PktLoss: \quad cwnd_{k+1} &= cwnd_k + \frac{1}{cwnd_k} \end{aligned} \tag{2}$$

where the delay component is computed as:

$$\begin{aligned} AckRec: dwnd_{k+1} = & \; dwnd_k + \alpha dwnd_k^K - 1, \text{ if } diff < \gamma \\ & dwnd_k - \eta diff, \quad \text{ if } diff \geq \gamma \\ PktLoss: dwnd_{k+1} = & \; dwnd_k(1-\beta) - \frac{cwnd_k}{2} \end{aligned} \tag{3}$$

where α, β, η and K parameters are chosen as a tradeoff between responsiveness, smoothness, and scalability.

Compound TCP dynamics is often dominated by its loss based component. Hence, it presents a slow responsiveness to network available bandwidth variations, which may cause playout buffer underflows.

D. Capacity and Congestion Probing TCP

TCP CCP was our first proposal of a delay based congestion avoidance scheme based on solid control theoretical approach. The cwnd size is adjusted according to a proportional controller control law. The cwnd adjustment scheme is called at every acknowledgement reception, and may result in either window increase or decrease. In addition, packet loss does not trigger any special cwnd adjustment. CCP cwnd adjustment scheme is as per (4):

$$cwnd_k = \frac{[Kp(B - x_k) - in\_flight\_segs_k]}{2} \quad 0 \leq Kp \qquad (4)$$

where Kp is a proportional gain, B is an estimated storage capacity of the TCP session path, or virtual buffer size, x_k is the level of occupancy of the virtual buffer, or estimated packet backlog, and in_flight_segs is the number of segments in flight (unacknowledged). Typically, CCP cwnd dynamics exhibit a dampened oscillation towards a given cwnd size, upon cross traffic activity. Notice that $cwnd_k$ does not depend on previous cwnd sizes, as with the other TCP variants. This fact guarantees a fast responsiveness to network bandwidth variations.

E. Linked Increase Congestion Control

Link Increase Algorithm [11] couples the congestion control algorithms of different sub-flows by linking their congestion window increasing functions, while adopting the standard halving of $cwnd$ window when a packet loss is detected. More specifically, LIA cwnd adjustment scheme is as per (5):

$$\begin{aligned} AckRec : cwnd^i_{k+1} &= cwnd^i_k + min(\frac{\alpha B_{ack} Mss^i}{\sum_0^n cwnd^i}, \frac{B_{ack} Mss^i}{cwnd^i}) \\ PktLoss : cwnd^i_{k+1} &= cwnd^i_k + \frac{1}{cwnd^i_k} \end{aligned} \qquad (5)$$

where α is a parameter regulating the aggressiveness of the protocol, B_{ack} is the number of acknowledged bytes, Mss^i is the maximum segment size of sub-flow i, and n is the number of sub-flows. Equation (5) adopts $cwnd$ in bytes, rather than in packets (MSS), in contrast with previous TCP variants, because now we have the possibility of diverse MSSs on different sub-flows. However, the general idea is to increase $cwnd$ in increments that depend on $cwnd$ size of all sub-flows, for fairness, but no more than a single TCP Reno flow. The min operator in the increase adjustment guarantees that the increase is at most the same as if MPTCP was running on a single TCP Reno sub-flow. Therefore, in practical terms, at each sub-flow LIA increases $cwnd$ at a slower pace than TCP Reno, still cutting $cwnd$ in half at each packet loss.

V. Video Streaming Performance of Congestion Avoidance Schemes

Fig. 3 describes the network testbed used for emulating a network path with wireless access link. An HTTP video server is connected to two access switches which are connected to a link emulator, used to adjust path delay and inject controlled random packet loss. A VLC client machine is connected to two Access Points, a 802.11a and 802.11g, on different bands (5GHz and 2.4GHz, respectively). All wired links are 1Gbps. No cross traffic is considered, as this would make it difficult to isolate the impact of TCP congestion avoidance schemes on video streaming performance. The simple topology and isolated traffic allows us to better understand the impact of differential delays on streaming performance.

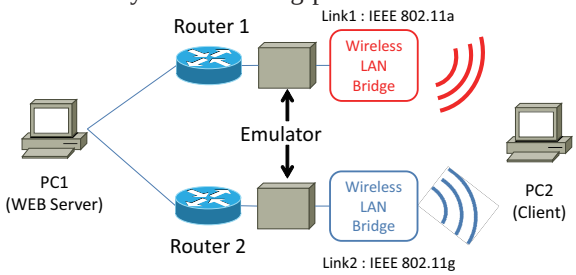


Figure 3: Video Streaming Emulation Network

TCP variants used are: Cubic, Compound, CCP, and LIA. Performance is evaluated for various round trip time path scenarios, as per Table I.

Table I: EXPERIMENTAL NETWORK SETTINGS

Element	Value
Video size	409Mbytes
Video rate	5.24Mbps
Playout time	10mins 24 secs
Encoding	MPEG-4
Video Codec	H.264/AVC
Audio Codec	MPEG-4 AAC4
Network Delay (RTT)	3, 50, 100 msecs
TCP variants	Cubic, Compound, CCP, LIA

The VLC client is attached to the network via a WiFi link. Iperf is used to measure the available wireless link bandwidth. UDP traffic injection experiments show that each wireless interface is limited to 5Mbps download speeds, which is lower than the video nominal playout rate of 5.24Mbps. Packet loss is hence induced only by the wireless link, and is reflected in the number of TCP packet retransmissions.

Performance measurers adopted, in order of priority, are:

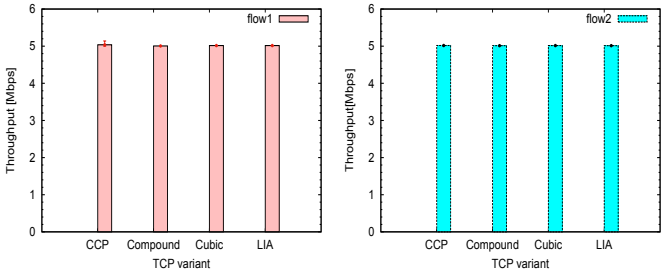
- **Picture discards:** number of frames discarded by the video decoder. This measurer defines the number of frames skipped by the video rendered at the client side.
- **Buffer underflow:** number of buffer underflow events at video client buffer. This measurer defines the number of "catch up" events, where the video freezes and then resumes at a faster rate until all late frames have been played out.
- **Packet retransmissions:** number of packets retransmitted by TCP. This is a measure of how efficient the TCP variant is in transporting the video stream data. It is likely to impact video quality in large round trip time path conditions, where a single retransmission doubles network latency of packet data from an application perspective.

 ISBN: 978-1-61208-516-6

We organize our video streaming experimental results into the following sub-sessions: i) Single path delay; ii) Equal path delay; iii) Differential path delay. Each data point in charts represents five trials. Results are reported as average and min/max deviation bars.

A. Single Path Video Streaming Performance Evaluation

Fig. 4 reports on video streaming throughput performance over a single path, under short propagation delay of 3msec. The figure shows throughput over the path through Router 1 (a), and path through Router 2 (b), respectively. In this case, all TCP variants suffer from a shortage of wireless download bandwidth, as indicated by the throughput of less than 5Mbps, below the average playout rate of 5.24 Mbps. This causes the streaming session last for tens of minutes or more, regardless of how much the path delays are.

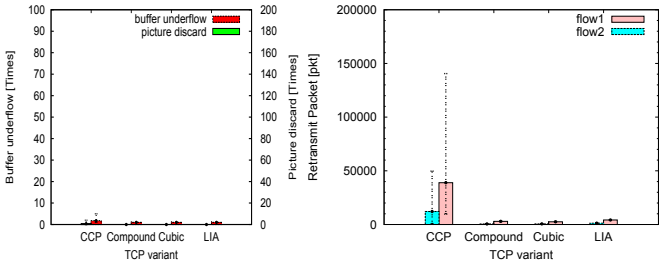


a) Flow 1 throughput b) Flow 2 throughput

Figure 4: One Path Streaming Throughput Performance; rtt=3msec

B. Equal Path Video Streaming Performance Evaluation

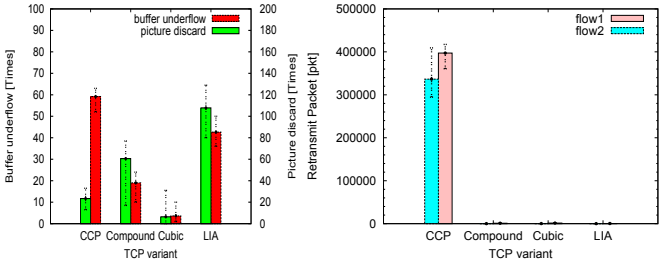
Fig. 5 reports on video streaming and MPTCP performance under short propagation delay of 3msec. In this case, all TCP variants deliver similar video streaming performance, with negligible number of frame discards and buffer underflow events. At the transport layer, we see that CCP presents a large number of retransmissions, in contrast with the other TCP variants, due to its aggressiveness and lack of reaction to random packet losses of the wireless links.



a) VLC performance b) TCP packets retransmitted

Figure 5: Equal Path Streaming Performance; rtt=3msec

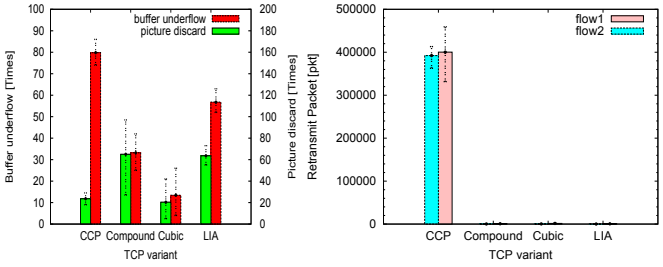
Fig. 6 reports on video streaming and TCP performance under a typical propagation delay of 50msec. In this case, Cubic delivers best video experience, with fewest picture discards and buffer underflows. Our CCP delivers second best picture discard performance, while presenting the highest buffer underflow event count, followed by LIA. We believe that a high TCP level retransmission rate causes packets to be held back at the TCP socket, causing video playout buffer to empty out multiple times. Compound TCP presents almost three times as much picture discards as CCP.



a) VLC performance b) TCP packets retransmitted

Figure 6: Equal Path Streaming Performance; rtt=50msec

Fig. 7 reports on video streaming and TCP performance under a large propagation delay of 100msec. Delays such as that may be experienced in paths with cellular network access links, where additional delays result from wireless access link level retransmissions. In this case, legacy TCP variant Cubic delivers best video performance overall. CCP presents a similar number of picture discards as Cubic, but with largest video buffer underflow event count, again due to large TCP level retransmissions. LIA and Compound TCP present the worst video performance.



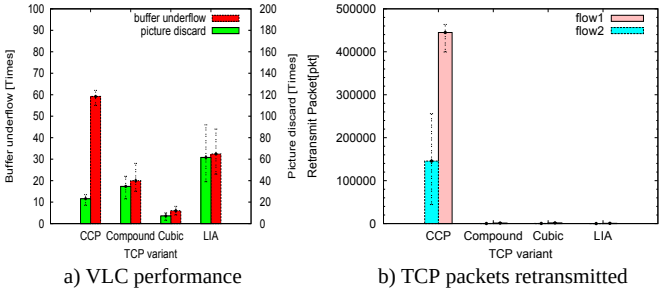
a) VLC performance b) TCP packets retransmitted

Figure 7: Equal Path Streaming Performance; rtt=100msec

C. Differential Path Video Streaming Performance Evaluation

In these scenarios, MPTCP scheduler tend to select the path with shorter delay. Only when TCP sender of the path with shorter delay happens to set its $cwnd$ to a very low value as compared with the longer path does MPTCP scheduler inject packets into the longer path.

Fig. 8 reports on video streaming and TCP performance under two paths, the first path (802.11a) with 50msec delay, and the other (802.11g) with 100msec delay. The relative performance of TCP variants is the same as in the previous equal path case. Cubic delivers best performance, followed by CCP, Compound, and LIA. The same high level packet retransmissions is incurred by CCP, not present in other variants.

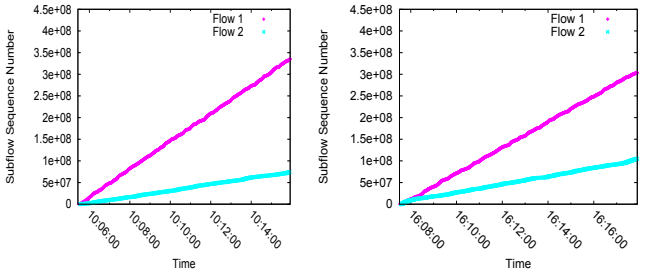


a) VLC performance b) TCP packets retransmitted

Figure 8: Differential Path Streaming Performance; rtt=50,100msec

 ISBN: 978-1-61208-516-6

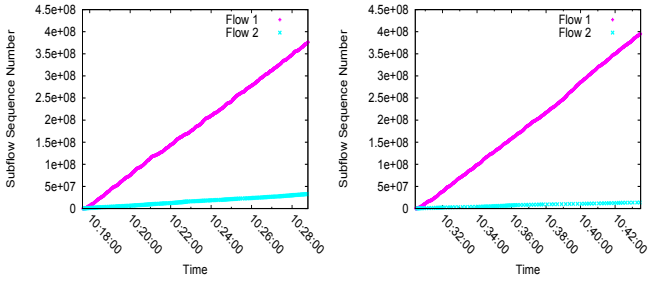
We have also tracked video streaming and TCP performance with path delay values swapped as compared with previous case: 802.11a path with 50msec delay, and 802.11g path with 100msec delay. The relative performance of TCP variants remains the same as in the previous case. To understand why, we monitored path utilization by tracking sub-flow sequence numbers. Fig. 9 plots Cubic TCP session sequence number dynamics of a video stream for a differential delay of 50 msecs. Figs. 9 a) and b) show reversed path differential cases. Notice that flow 1 always presents higher SN slope, due to the fact that path 2 wireless link has less bandwidth than path 1, and Cubic adjusts to it by reducing flow 2 $cwnd$ much further than flow 1 $cwnd$. The amount of differential delay also impacts utilization of path 2. In addition, SN progressing is steady, which means that MPTCP scheduler keeps distributing packets across both paths throughout the video session.



a) p1: 50msec; p2: 100msec b) p1: 100msec; p2: 50msec

Figure 9: Cubic Seq. Num. Dynamics; rtt=50,100msec

About sub-flow utilization, Fig. 10 reports on LIA and Compound TCP variants sub-flow sequence number dynamics. Notice how little LIA utilizes path 2, whereas Compound uses it a little more, but not as much as Cubic.



a) Compound TCP b) LIA TCP

Figure 10: Compound/LIA Seq. Num. Dynamics; rtt=50,100msec

In conclusion, although being the recommended TCP variant for MPTCP, LIA delivers worst video performance than when it operates in uncoupled mode with Cubic, Compound, and CCP TCP variants for various dual path settings. LIA also fails to push more traffic on less bandwidth paths. In our extensive real time testbed results, Cubic is the clear winner in both delivering best video streaming over two paths as well as balancing traffic load. Our CCP TCP variant comes in second, albeit suffering from a large retransmission issue which causes a significant number of buffer underflow events.

In our performance evaluation, we have not attempted to tune VLC client to minimize frame discards, even though VLC settings may be used to lower the number of frame discards. In addition, no tuning of TCP parameters was performed to better video client performance.

VI. Conclusion and Future Work

In this paper, we have evaluated Multipath TCP transport of video streaming, using widely deployed TCP variants, as well as LIA coupled TCP variant currently under consideration by IETF. We have characterized MPTCP performance with these TCP variants when transporting video streaming over two wireless network paths via open source experiments. Our experimental results show that Cubic delivers best streaming performance, with fewer picture discards and less video stalls, across a wide range of path round trip times. As more complex network scenarios present both limited bandwidth paths as well as differential path delays, we expect similar impact of these impairments on video streaming performance.

As MPTCP scheduler switches frequently between paths, driven by $cwnd$ and path delay changes, triggering buffer underflow events due to frame reordering at the receiver. we are currently studying schemes to reduce buffer underflow events, especially when path delays are significantly different. We also intend to explore different MPTCP coupling schemes.

Acknowledgments

This work is supported by JSPS KAKENHI Grant Number 16K00131.

References

[1] M. Allman, V. Paxson, and W. Stevens, “TCP Congestion Control,” IETF RFC 2581, April 1999.

[2] D. Cavendish, K. Kumazoe, M. Tsuru, Y. Oie, and M. Gerla, “Capacity and Congestion Probing: TCP Congestion Avoidance via Path Capacity and Storage Estimation,” IEEE Second International Conference on Evolving Internet, best paper award, pp. 42-48, September 2010.

[3] D. Cavendish, H. Kuwahara, K. Kumazoe, M. Tsuru, and Y. Oie, “TCP Congestion Avoidance using Proportional plus Derivative Control,” IARIA Third International Conference on Evolving Internet, best paper award, pp. 20-25, June 2011.

[4] H. Ishizaki et al., “On Tuning TCP for Superior Performance on High Speed Path Scenarios,” IARIA Fourth International Conference on Evolving Internet, best paper award, pp. 11-16, June 2012.

[5] G. Watanabe et al., “Performance Characterization of Streaming Video over TCP Variants,” IARIA Fifth International Conference on Evolving Internet, best paper award, pp. 16-21, June 2013.

[6] G. Watanabe et al., “Slow Start TCP Improvements for Video Streaming Applications,” IARIA Sixth International Conference on Evolving Internet, best paper award, pp. 22-27, June 2014.

[7] D. Cavendish et al., “Congestion Avoidance TCP Improvements for Video Streaming,” IARIA Seventh International Conference on Evolving Internet, best paper award, pp. 22-27, October 2015.

[8] X. Corbillon, R. Aparicio-Pardo, N. Kuhn, G. Texier, and G. Simon, “Cross-Layer Scheduler for Video Streaming over MPTCP,” ACM 7th International Conference on Multimedia Systems, May 10-13,2016, Article 7.

[9] A. Ford, et al., “Architectural Guidelines for Multipath TCP Development,” IETF RFC 6182, 2011.

[10] J-W. Park, R. P. Karrer, and J. Kim,, “TCP-Rome: A Transport-Layer Parallel Streaming Protocol for Real-Time Online Multimedia Environments,” In Journal of Communications and Networks, Vol.13, No. 3, pp. 277-285, June 2011.

[11] C. Raiciu, M. Handly, and D. Wischik, “Coupled Congestion Control for Multipath Transport Protocols,” IETF RFC 6356, 2011.

[12] I. Rhee, L. Xu, and S. Ha, “CUBIC for Fast Long-Distance Networks,” Internet Draft, draft-rhee-tcpm-ctcp-02, August 2008.

[13] M. Sridharan, K. Tan, D. Bansal, and D. Thaler, “Compound TCP: A New Congestion Control for High-Speed and Long Distance Networks,” Internet Draft, draft-sridharan-tcpm-ctcp-02, November 2008.

[14] J. Wu, C. Yuen, B. Cheng, M. Wang, and J. Chen, “Streaming High-Quality Mobile Video with Multipath TCP in Heterogeneous Wireless Networks,” IEEE Transactions on Mobile Computing, to be published.

 ISBN: 978-1-61208-516-6