TCP State Driven MPTCP Packet Scheduling for Streaming Video

Ryota Matsufuji, Shinichi Nagayama, Dirceu Cavendish, Daiki Nobayashi, Takeshi Ikenaga
Department of Computer Science and Electronics
Kyushu Institute of Technology
Fukuoka, Japan
e-mail: {q349428r@mail, o108076s@mail}.kyutech.jp {cavendish@ndrc, nova@ecs, ike@ecs}.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 allowed video streaming over multiple paths to become a reality. In this paper, we propose packet scheduling disciplines driven by underline TCP flow state for injecting video stream packets into multiple paths at the video server. We show how video streaming performance improves when packet schedulers take into account retransmission state in underlying paths in conjunction with current TCP variants. We utilize network performance measures, as well as video quality metrics, to characterize the performance and interaction between network and application layers of video streams for various network scenarios.**

Keywords*—Video streaming; high speed networks; TCP congestion control; TCP socket state; 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 service 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.

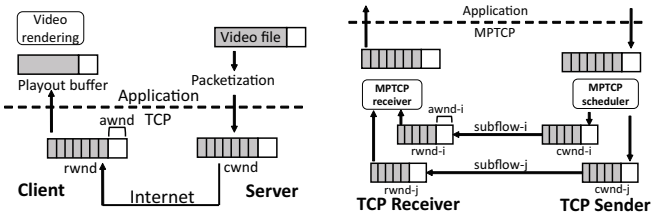
Recently, multipath transport has allowed video streamed over multiple IP interfaces and network paths. Multipath streaming not only augments aggregated bandwidth, but also increases reliability at the transport level session even when a specific radio link coverage gets compromised. An important issue in multipath transport is the path (sub-flow) selection; a packet scheduler is needed to split traffic to be injected on a packet by packet basis. For video streaming applications, head of line blocking may cause incomplete or late frames to be discarded at the receiver, as well as stream stalling. In this work, we introduce the concept of path schedulers based on current status of a TCP sub-flow and evaluate video streaming performance under this type of schedulers. To the best of our knowledge, there has not been a study of path selection mechanisms based on TCP sub-flow state. Specifically, we show that by avoiding paths in retransmission state, video streaming performance improvements can be obtained for different TCP variants and packet scheduler schemes.

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. Section V introduces path schedulers used to support multipath transport, as well as our new TCP state driven path scheduling proposal. Section VI addresses multiple path video delivery performance evaluation for each TCP variant and multiple packet schedulers. Our empirical results in that section show that most schedulers benefit from TCP state awareness. Section VII addresses directions we are pursuing as follow up to this work.

II. Related Work

Although multipath transport studies are plenty in the literature, there has been few prior work on video performance over multiple paths [5] [13] [16]. Regarding multipath schedulers, there has been limited research activity. Yan et al. [18] propose a path selection mechanism based on estimated sub-flow capacity. Their evaluation is centered on throughput performance, as well as reducing packet retransmissions. Yan et al. [2] present a modelling of multipath transport in which they explain empirical evaluations of the impact of selecting a first sub-flow in throughput performance. Hwang et al. [9] propose a blocking scheme of a slow path when delay difference between paths is large, in order to improve data transport completion time on short lived flows. Ferlin et al. [7] introduce a path selection scheme based on a predictor of the head-of-line blocking of a given path. They carry out emulation experiments with their scheduler against the minimum Round Trip Time (RTT) default scheduler, in transporting bulk data, Web transactions and Constant Bit Rate (CBR) traffic, with figure of merits of goodput, completion time and packet delays, respectively. More recently, Kimura et al. [11] have shown throughput performance improvements on schedulers driven by path sending rate and window space, focusing on bulk data transfer applications. Also, Dong et al. [6] have proposed a path loss estimation approach to select paths subject to high and bulk loss rates. Although they have presented some Video Streaming experiments, they do not measure streaming performance from an application perspective. Finally, [17] has proposed a path scheduler based on prediction of the amount of data a path is able to transmit and evaluated it on simulated network scenarios with respect to throughput performance.

 ISBN: 978-1-61208-644-6



(a) TCP (b) MPTCP

Figure 1: Video Streaming over TCP/MPTCP

In contrast, our current work seeks multipath path scheduling principles that can be applied to different path schedulers to improve the quality of video streams. Previously [12], we have proposed new Multipath TCP (MPTCP) path schedulers based on dynamic path characteristics, such as congestion window space and estimated path throughput and evaluated multipath video streaming using these proposed schedulers. In this work, we propose to enhance path schedulers with TCP state information, such as whether a path is in fast retransmit and fast recovery state, to improve video quality in lossy network scenarios. For performance evaluation, we focus on video stream applications and use widely deployed TCP variants on open source network experiments over WiFi access links.

III. Video Streaming over TCP

Video streaming over HTTP/TCP involves an HTTP server, 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. Figure 1 (a) illustrates video streaming components.

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, the 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 (AIMD) strategy, with various increase and decrease parameters. TCP NewReno [1] and Cubic [15] are examples of 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. Capacity and Congestion Probing (CCP) [3] and Capacity Congestion Plus Derivative (CCPD) [4] are examples of delay based protocols.

Most TCP variants follow TCP Reno phase framework: slow start, congestion avoidance, fast retransmit and fast recovery. For TCP variants widely used today, congestion avoidance phase is sharply different. We will be introducing specific TCP variants' congestion avoidance phase shortly.

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 [8]. 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 injects them into TCP sockets belonging to a "subflow" defined by a single path TCP session. MPTCP transport architecture is represented in Figure 1 (b).

 ISBN: 978-1-61208-644-6

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 with parameters of all sub-flows. Although there are several coupled mechanisms, we focus on Linked Increase Algorithm (LIA) [14] and Opportunistic Linked Increase Algorithm (OLIA) [10]. In both cases, a MPTCP scheduler selects a sub-flow for packet injection according to some criteria among all sub-flows with large enough $cwnd$ to allow packet injection.

B. Linked Increase Congestion Control

Link Increase Algorithm (LIA) [14] couples the congestion control algorithms of different sub-flows by linking their congestion window increasing functions, while adopting the standard halving of $cwnd$ window upon packet loss detection. More specifically, LIA cwnd adjustment scheme is as per (1):

$$\begin{aligned} AckRec : cwnd^i_{k+1} &= cwnd^i_k + min(\tfrac{\alpha B_{ack} Mss^i}{\sum_0^n cwnd^p}, \tfrac{B_{ack} Mss^i}{cwnd^i}) \\ PktLoss : cwnd^i_{k+1} &= \tfrac{cwnd^i_k}{2} \end{aligned} \quad (1)$$

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 (1) adopts $cwnd$ in bytes, rather than in packets (Maximum Segment Size - MSS), in contrast with TCP variants equations to be described shortly, because here 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, each LIA sub-flow increases $cwnd$ at a slower pace than TCP Reno, still cutting $cwnd$ in half at each packet loss.

C. Opportunistic Linked Increase Congestion Control

Opportunistic Link Increase Algorithm (OLIA) [10] also couples the congestion control algorithms of different sub-flows, but with the increase based on the quality of paths. OLIA cwnd adjustment scheme is as per (2):

$$\begin{aligned} AckRec : cwnd^i_{k+1} &= cwnd^i_k + \frac{\frac{cwnd^i}{(RTT^i)^2}}{(\sum_0^n \frac{cwnd^p}{RTT^p})^2} + \frac{\alpha^i}{cwnd^i}, \\ PktLoss : cwnd^i_{k+1} &= \tfrac{cwnd^i_k}{2} \end{aligned} \quad (2)$$

where α is a positive parameter for all paths. The general idea is to tune $cwnd$ to an optimal congestion balancing point (in the Pareto optimal sense). In practical terms, each OLIA sub-flow increases $cwnd$ at a pace related to the ratio of its RTT and RTT of other subflows, still cutting $cwnd$ in half at each packet loss.

D. 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:

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

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). Parameters K drives the cubic increase away from Wmax, whereas β tunes how quickly cwnd reduction happens on packet loss. This process recovers its $cwnd$ quickly after causing loss event.

E. 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 regardless of loss event. CCP cwnd adjustment scheme is as per (4):

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

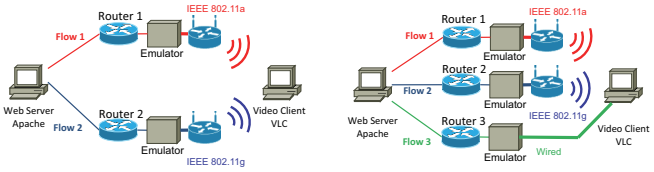
where Kp is a proportional gain, B is an estimated storage capacity of the TCP session path, or virtual buffer size, x_k is the level of occupancy of the virtual buffer, or estimated packet backlog and in_flight_segs is the number of segments in flight (unacknowledged). This fact guarantees a fast responsiveness to network bandwidth variations.

V. TCP State Driven MPTCP Packet Scheduler

MPTCP scheduler selects which sub-flow to inject packets into the network on a packet by packet basis. The default strategy is to select the path with shortest average packet delay. Herein, we introduce this conventional SPD, our previous LPC, LET path selection scheme, as well as a TCP state/retransmission aware packet injection mechanisms.

- **Shortest Packet Delay(SPD):** In shortest packet delay, the scheduler first rules out any path for which there is no space in its sub-flow congestion window (cwnd). Among the surviving paths, the scheduler then selects the path with small smooth round trip time (RTT). Smooth RTT is computed as an average RTT of recent packets transmitted at that sub-flow. Since each sub-flow already keeps track of its smooth RTT, this quantity is readily available at every sub-flow.
- **Largest Packet Credits(LPC):** Among the sub-flows with space in their cwnd, this scheduler selects the one with largest available space. Available space is the number of packets allowed by cwnd size minus the packets that have not been acknowledged yet.

 ISBN: 978-1-61208-644-6



(a) Two Path Network (b) Three Path Network

Figure 2: Video Streaming Emulation Network

- **Largest Estimated Throughput(LET):** In this case, among the sub-flows with large enough cwnd to accommodate new packets, the scheduler estimates the throughput of each sub-flow, as cwnd/sRTT (smooth RTT) and selects the one with largest throughput.
- **RTX Aware:** This supplemental scheme aims to avoid injecting packets into paths that are in retransmit/recovery mode, which would increase packet delivery delay due to head of line blocking. The strategy can be applied on top of any packet scheduler. In this work, it is applied to all previous schedulers (SPDX, LPCX and LETX, respectively). For instance, LETX first eliminates paths in TCP retransmission and among the remaining ones it selects the path of maximum estimated throughput. If all sub-flows are in retransmission state, no path is selected.

The rationale for these proposed schedulers is as follows. LPC addresses the path scenario in which a large RTT path has plenty of bandwidth. In default scheduler, this path may be less preferred due to its large RTT, regardless of having plenty of bandwidth for the video stream. LET addresses the scenario of a short path with plenty of bandwidth. The default scheduler may select this path due to its short RTT. However, if the short RTT has a smaller cwnd, LET will divert traffic away from this path, whereas default scheduler will continue to inject traffic through it. RTX Aware addresses network scenarios experiencing packet loss unevenly across multiple paths.

VI. Video Streaming Performance of Multipath Schedulers

Figure 2 describes the network testbeds used for emulating a network path with wireless and wired access links. On the first testbed, an HTTP Apache video server is connected to two access switches, which are connected to link emulators, 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). On the second testbed, one extra all wired network path is added between the video server and the VLC client. 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 topologies and isolated traffic allows us to better understand the impact of differential delays on streaming performance.

We list network settings and scenarios generated by network emulator in Tables I and II, respectively. Video settings are typical of a video stream. Its size is short enough to enable multiple streaming trials within a reasonable amount of time.

TABLE I: EXPERIMENTAL NETWORK SETTINGS

Element	Value
Video size	409 MBytes
Video rate	5.24 Mbps
Playout time	10 mins 24 secs
Video Codec	H.264 MPEG-4 AVC
MPTCP variants	CCP, Cubic, LIA, OLIA
MPTCP schedulers	SPD, LPC, LET, SPDX (rtX aware SPD), LPCX, LETX

TABLE II: EXPERIMENTAL NETWORK SCENARIO

Scenario	Emulator configuration (RTT, Bandwidth, Random loss rate)
3 path Equal Delay (3p-e)	Flow1) RTT 100 ms, BW 3 Mb/s, Loss 0 % Flow2) RTT 100 ms, BW 3 Mb/s, Loss 0 % Flow3) RTT 100 ms, BW 3 Mb/s, Loss 0.5 %
3 path Differential Delay (3p-d)	Flow1) RTT 100 ms, BW 3 Mb/s, Loss 0 % Flow2) RTT 100 ms, BW 3 Mb/s, Loss 0 % Flow3) RTT 50 ms, BW 3 Mb/s, Loss 0.5 %
2 path Equal Delay (2p-e)	Flow1) RTT 100 ms, BW 5 Mb/s, Loss 0.5 % Flow2) RTT 100 ms, BW 5 Mb/s, Loss 0 %
2 path Differential Delay (2p-d)	Flow1) RTT 50 ms, BW 5 Mb/s, Loss 0.5 % Flow2) RTT 100 ms, BW 5 Mb/s, Loss 0 %

For each scenario, path bandwidth capacity is tuned so as to force the use of multiple paths to stream a video playout rate of 5.24Mbps. We also inject 0.5 packet loss rate on the shortest path of each scenario, so as to contrast default packet scheduler (shortest RTT) with other schedulers. TCP variants used are: CCP, Cubic, LIA and OLIA.

Performance measures adopted, in order of priority, are:

- **Picture discards:** number of frames discarded by the video decoder. This measure 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 measure 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.
- **Recovery Time from underflow:** amount of time a video playout buffer remains empty after an underflow event. This measure defines how long it takes for underflow event to recover and start rebuffering application data.
- **Sub-flow throughput:** the value of TCP Throughput on each sub-flow. This measure captures how MPTCP operates its scheduling packet injection and whether it is able to maintain a high enough throughput for the video playout rate.

We organize our video streaming experimental results in two network scenarios: i) Two path MPTCP; ii) Three path MPTCP. Each data point in charts represents five trials. Results are reported as average and min/max deviation bars.

A. Two Path MPTCP Performance Evaluation

Figures 3 a, b, c, d, report on video streaming and TCP performance in scenario 2p-e, 2path equal delay and a lossy path. For CCP variant (a), there is a small perceivable video performance (picture discard/buffer underflow) improvement by using RTX awareness on all schedulers. For Cubic TCP variant (b), there is a significant video performance improvement when RTX awareness is used in LPC, whereas LET seems to get worst. On the other hand, LIA and OLIA TCP variants (c,d) provide an appreciable video performance improvement when RTX awareness is used with all schedulers.

 ISBN: 978-1-61208-644-6

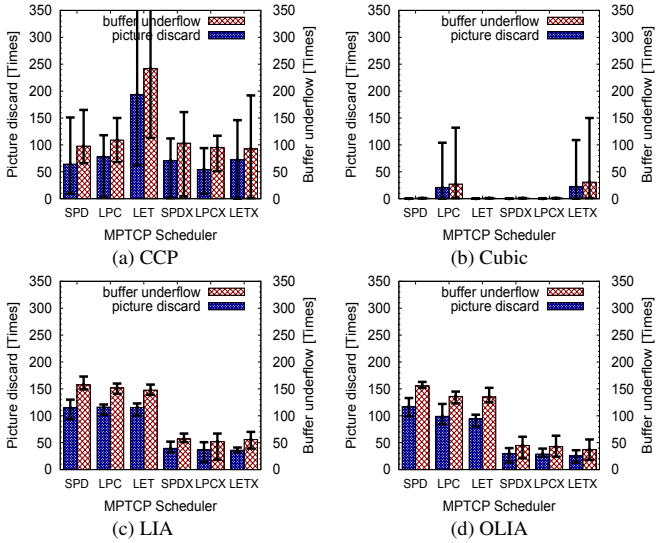


Figure 3: Scheduler Streaming Perf.; Scenario 2p-e

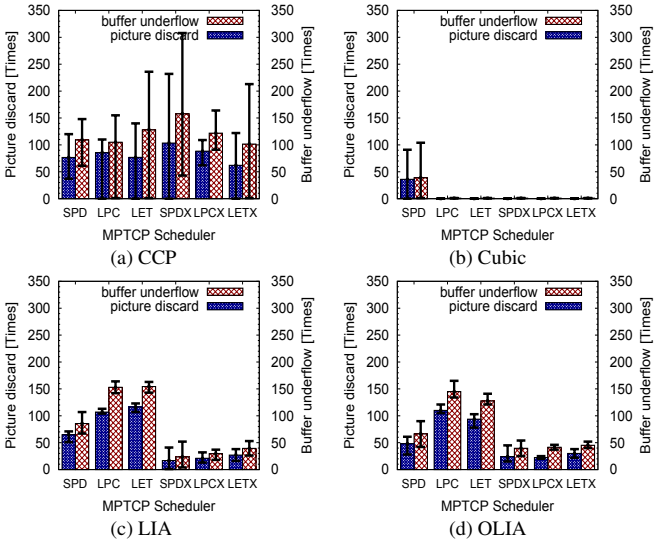


Figure 4: Scheduler Streaming Perf.; Scenario 2p-d

Figures 4 a, b, c, d, describe video streaming performance under network scenario 2p-d, where 0.5 % random packet loss is injected into the shorter delay path. Notice that the SDP scheduler gives preference to shorter delay path, regardless of its packet loss, which hurts performance. For CCP variant (a), there is no perceivable video performance difference among all schedulers. That is because CCP congestion avoidance often suffers from inaccurate estimation of path capacity. In contrast, Cubic often works well for video streaming independently of packet scheduler. On the other hand, coupled LIA and OLIA deliver best video performance when adopting RTX Aware strategy over all schedulers, while non-RTX schedulers cause a lot of video error events. In addition, Figures 5 a, b, c, d, report on corresponding recovery time of each scheduler and TCP variant. We can see that retransmission aware scheduling allows video client to refill quickly video receiver buffer, especially for LIA/OLIA TCP variants. There seems to be little impact on recovery time for more aggressive Cubic/CCP variants, due to their aggressive congestion window ramp up.

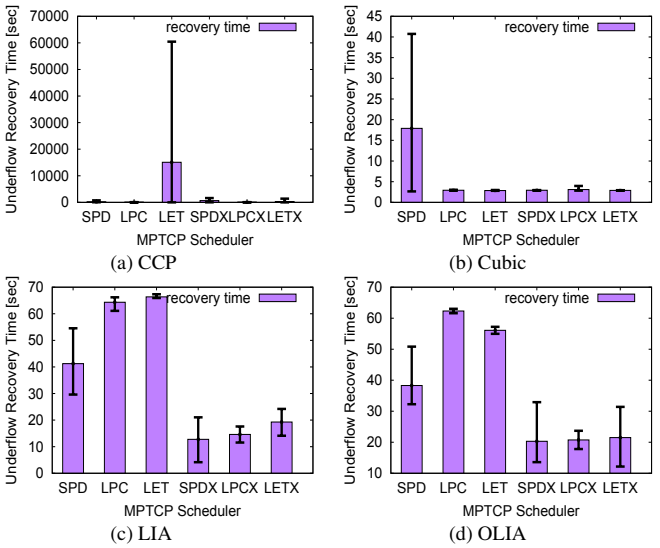


Figure 5: Scheduler Recovery Time.; Scenario 2p-d

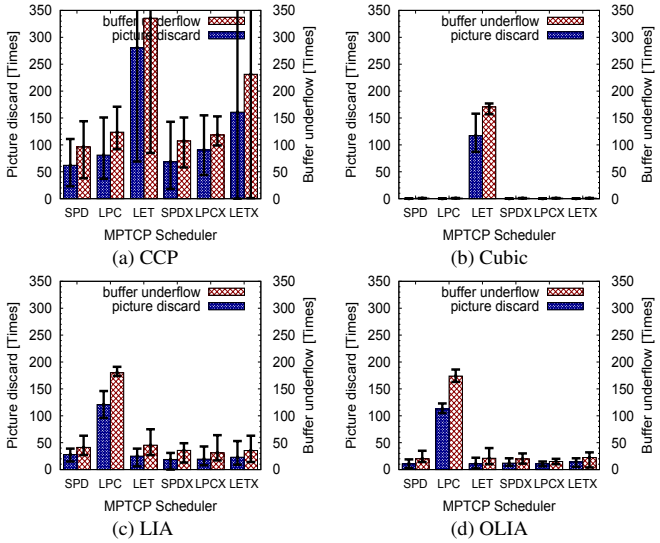
B. Three Path MPTCP Performance Evaluation

Figures 6 a, b, c, d, show video streaming and TCP performance under scenario 3p-e, three path equal delay RTT 100 msec with a 0.5 % random lossy path. In Figure 6 (a), no scheduler is able to improve CCP to deliver high video playout performance in 3 path network scenario. This is because CCP underestimates $cwnd$ in lossy and narrow bandwidth paths. Cubic (6 (b)), on the other hand, delivers best video performance under SPD and LPC schedulers. In addition, RTX Aware strategy increases LET video performance significantly. In contrast, RTX Aware strategy for LIA and OLIA decreases discard/underflow events when LPC scheduler is used.

Finally, Figures 7 a, b, c, d present video performance in scenario 3p-d, where shortest RTT flow3 has a 0.5 % packet loss condition. CCP and Cubic charts are similar as in previous scenario 3p-e, namely, little performance improvement by changing packet scheduler except for LET scheduler under Cubic TCP variant, which presents significant improvement. LIA and OLIA schedulers (Figures 7 c,d), on the other hand, provide only small improvements when RTX awareness is used. Figures 8 (c,d) shows that LET scheme injects a larger amount of packets into loss-less flow1 and flow2 than SPD and LPC, since total bandwidth of loss-less flow1 and flow2 is capable of 5.24 Mb/s video traffic in scenario 3p-e.

Overall, the above results show a consistent video streaming performance improvement when paths experiencing momentary retransmissions are avoided across most TCP variants as well as path scheduler schemes. In addition, more available paths does not always bring better performance, especially for aggressive TCP variants such as Cubic and CCP. Although these results were obtained for specific testbed topologies and network scenarios, we believe similar improvements can be attained on more generic network scenarios.

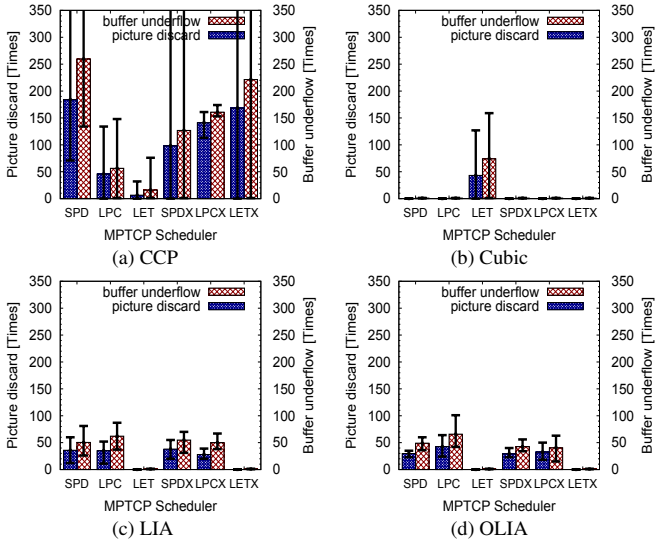
 ISBN: 978-1-61208-644-6



(a) CCP (b) Cubic

(c) LIA (d) OLIA

Figure 6: Scheduler Streaming Perf.; Scenario 3p-e



(a) CCP (b) Cubic

(c) LIA (d) OLIA

Figure 7: Scheduler Streaming Perf.; Scenario 3p-d

VII. Conclusion and Future Work

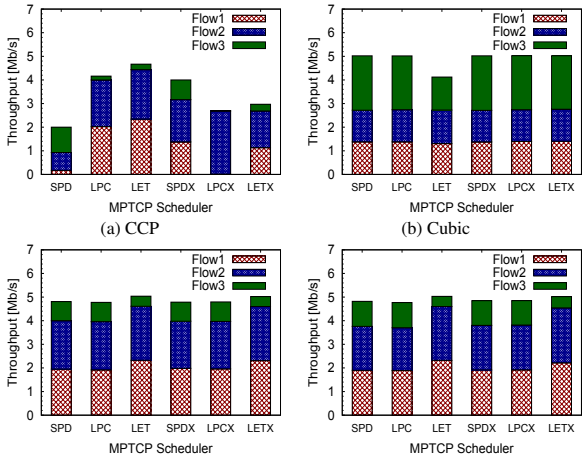
In this paper, we have proposed TCP state driven packet schedulers to improve the quality of streaming video over MPTCP. We have evaluated MPTCP performance with default and several packet schedulers which avoid injecting packets into paths experiencing retransmissions in lossy wireless network. Our results have shown that TCP state aware scheduling improves video streaming across most TCP variants, as well as coupled LIA and OLIA, for all packet schedulers studied. We believe that avoiding paths experiencing packet retransmissions may be applicable across a wide variety of schedulers and TCP variants. We are currently investigating other scheduling techniques to further reduce frame discard and video stalling to improve streaming video quality.

Acknowledgments

Work supported by JSPS KAKENHI Grant # 16K00131.

References

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



(a) CCP (b) Cubic

(c) LIA (d) OLIA

Figure 8: Scheduler Throughput Perf.; Scenario 3p-d

[2] B. Arzani et al., "Deconstructing MPTCP Performance," In Proceedings of IEEE 22nd ICNP, pp. 269-274, 2014.

[3] 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, pp. 42-48, September 2010.

[4] 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, pp. 20-25, June 2011.

[5] 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.

[6] E. Dong et. al., "LAMPS: A Loss Aware Scheduler for Multipath TCP over Highly Lossy Networks," *Proceedings of the 42th IEEE Conference on Local Computer Networks*, pp. 1-9, October 2017.

[7] S. Ferlin et. al., "BLEST: Blocking Estimation-based MPTCP Scheduler for Heterogeneous Networks," In Proceedings of IFIP Networking Conference, pp. 431-439, 2016.

[8] A. Ford et. al., "Architectural Guidelines for Multipath TCP Development," IETF RFC 6182, 2011.

[9] J. Hwang and J. Yoo, "Packet Scheduling for Multipath TCP," IEEE 7th Int. Conference on Ubiquitous and Future Networks, pp.177-179, July 2015.

[10] R. Khalili, N. Gast, and J-Y Le Boudec, "MPTCP Is Not Pareto-Optimal: Performance Issues and a Possible Solution," IEEE/ACM Trans. on Networing, Vol. 21, No. 5, pp. 1651-1665, Aug. 2013.

[11] B. Kimura et al., "Alternative Scheduling Decisions for Multipath TCP," IEEE Communications Letters, Vol. 21, No. 11, pp. 2412-2415, Nov. 2017.

[12] R. Matsufuji et al., "Multipath TCP Packet Scheduling for Streaming Video," IEEE Pacific Rim Conference on Communications, Computers and Signal Processing, pp. 1-6, August 2017.

[13] 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.

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

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

[16] 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, Vol.15, Issue 9, pp. 2345-2361, 2016.

[17] K. Xue et al., "DPSAF: Forward Prediction Based Dynamic Packet Scheduling and Adjusting With Feedback for Multipath TCP in Lossy Heterogeneous Networks," IEEE/ACM Trans. on Vehicular Technology, Vol. 67, No. 2, pp. 1521-1534, Feb. 2018.

[18] F. Yan, P. Amer, and N. Ekiz, "A Scheduler for Multipath TCP," In Proceedings of IEEE 22nd ICCCN, pp. 1-7, 2013.

 ISBN: 978-1-61208-644-6