Handoff Characterization of Multipath Video Streaming

Handoff Characterization of Multipath Video Streaming

Kazuya Fujiwara, Shinichi Nagayama, Dirceu Cavendish, Daiki Nobayashi, Takeshi Ikenaga
Department of Computer Science and Electronics
Kyushu Institute of Technology
Fukuoka, Japan
e-mail: {q108107k@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 utilize 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 streaming of video over multiple paths. In this paper, we analyze the impact of handoffs on multipath video streaming and network performance on WiFi and cellular paths. We utilize network performance measures, as well as video quality metrics, to characterize the performance and interaction between network and application layers of video data 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) has become the most widely deployed transport protocol of the Internet, providing reliable data transmission for the overwhelming majority of applications. For data applications, the perceived quality of service can be summarized as the total transport time of a given file. For streaming applications, the perceived quality of experience involves the amount of data discarded at the client due to excessive transport delays, as well as rendering stalls due to lack of timely playout data. These performance measures, namely transport delays and data starvation, depend on how TCP handles flow control and packet retransmissions.

Motivated by the evolution of multiple device interfaces, multipath transport has been developed, allowing video streaming over multiple IP interfaces and network paths. Multipath streaming not only increases aggregated bandwidth capacity, but also increases reliability at the transport level session when a specific radio link coverage gets compromised. Moreover, an important issue in multipath transport is the path (sub-flow) selection; a path scheduler is needed to split traffic to be injected on a packet by packet basis onto available paths. Head of line blocking across different paths may cause incomplete or late frames to be discarded at the receiver, as well as stream stalling, compromising video rendering performance. In this work, we analyze the effect of path handoffs from a primary path to a secondary path on the quality of video stream delivery. As streaming session lasts long enough to experience path disconnection in many use cases, such as WiFi to Cellular handoffs, it is important to study such events from an application performance viewpoint.

The material is organized as follows. Related work is discussed on Section II. Section III details how video streaming is supported over TCP transport protocol. Section IV introduces widely deployed TCP variants utilized as transport for each path. Section V characterizes handoff effects on multiple path video delivery via WiFi and cellular paths via network emulation, addressing performance evaluation using a default path scheduler and a recently proposed sticky scheduler, for each TCP variant. Our empirical results show that Video streaming using coupled TCP variants may be impacted by handoffs, particularly on WiFi-Cellular scenarios. Section VI addresses directions we are pursuing as follow up to this work.

II. Related Work

Although there have been several multipath transport studies in the literature, few have focused on video performance over multiple paths. In what follows, we classify these efforts according to their scope, and comment on representative ones.

A. Multipath Video streaming on ad-hoc networks

These works are motivated by vehicular communication use cases emerging for assisted driving systems. A representative research effort within this scope is [2], which proposes an interference aware multipath video streaming in Vehicular Ad-hoc Networks (VANETs). They consider vehicle interference within neighbors, as well as shadowing effects onto Signal to Noise ratio, data delay and throughput of video streams over multiple paths. They also provide a good survey of recent work on multipath video streaming over VANETs. From a scope's perspective, even though the ultimate objective is reliable transport of high quality video streams, minimizing video freezes and dropped frames, these efforts are link layer approaches, such as channel interference, coupled with efficient routing strategies on ad-hoc vehicular networks. In contrast, our scope is video streaming over regular Internet, where channel and route optimization opportunities are limited.

B. Application driven path selection on heterogeneous paths

The scope here is in coupling application layer with transport protocol to increase video streaming quality. For instance, [1] proposes a path-and-content-aware path selection approach to couple MPEG Media Transport (MMT) protocol with multipath transport protocol. They estimate path quality condition of each subflow, and selectively avoid sending I-frames on paths of low quality. They evaluate video layer quality via Peak Signal to Noise (PSNR) tracing, as well as network layer goodput. A similar approach, at which different sub-flows are used for segregating prioritized packets of Augmented Reality/Virtual Reality streams has been proposed by Silva et al.

[21]. In contrast, our previous and current work do not couple application with multipath transport, as the coupling would require different transport protocols for different applications.

C. Multipath path selection of data transport within MPTCP

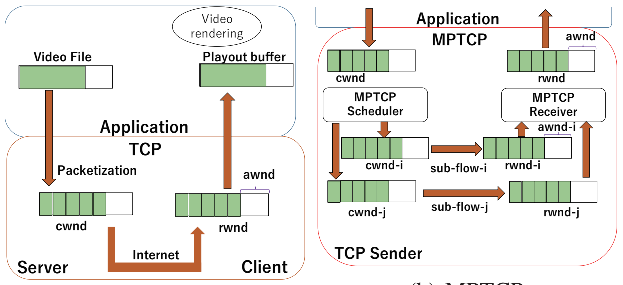
Here, the scope is smart path selection via sub-flow transport chanracterization. Arzani et al. [4] present a modelling of multipath transport in which they explain empirical evaluations of the impact of selecting a first sub-flow in throughput performance. Hwang et al. [10] propose a blocking scheme, where a slow path is not used when delay difference between paths is large, to improve data transport completion time on short lived flows. Ferlin et al. [7] introduce a path selection scheme based on a head-of-line blocking predictor of paths. They carry out emulation experiments of their scheduler against minimum Round Trip Time (RTT) default scheduler, in transporting bulk data, Web transactions and Constant Bit Rate (CBR) traffic. Performance evaluation metrics are goodput, completion time and packet delays, respectively.

More recently, Kimura et al. [12] have shown throughput performance improvements on schedulers driven by path sending rate and window space, focusing on bulk data transfer applications. Xue et al. [23] 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. Also, Frommgen et al. [9] have shown that stale round trip time (rtt) information interferes with path selection of small streams such as HTTP traffic. The authors propose an rtt probing and one way delay based path selection to improve latency and throughput performance of thin streams. Finally, [22] has addressed the WiFi/Cellular(LTE) handoff scenario when transferring data over MPTCP. They propose a radio/transport cross-layer approach, where TCP layer receives indication of a threshold SNR event crossing, indicating likely handoff. Via simulations, they show transport layer (throughput, RTT, retransmissions) improvements when WiFi/LTE handoffs occur, for Reno, Lia and Olia TCP variants on data transfers. In contrast, our handoff characterization focuses on impact of handoffs on video streaming quality.

D. Multipath path selection of Video Streams within MPTCP

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.

By contrast, in our previous work, we have proposed multipath path scheduling principles that can be applied to different path schedulers to specifically improve the quality of video streams. In [13], we have proposed Multipath TCP 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. Recently [14], we have also proposed to enhance path schedulers with TCP state information, such as whether a path is in fast retransmit and fast recovery, to improve



(a) TCP (b) MPTCP

Figure 1: Video Streaming over TCP/MPTCP

video quality in lossy network scenarios. In [15], we have introduced the concept of a sticky scheduling, where once a path switch occurs, we stay with the new path until its bandwidth resources become exhausted. In this work, we have included sticky scheduler as part of our handoff performance evaluation using widely deployed TCP variants on open source network experiments over WiFi and cellular paths. We focus on most commonplace scenario of handoffs between WiFi and cellular networks on video streaming sessions originated at home and lasting way after the user leaves its WiFi network.

III. Video Streaming Over TCP

At application layer, a video streaming over HTTP/TCP typically uses 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. At transport layer, a TCP variant is used to store and reliably transport video data over IP packets between the two end points. Figure 1 (a) illustrates video streaming components. The HTTP server stores encoded video files, making them available upon HTTP requests. Upon HTTP video request, a TCP sender is instantiated to transmit packetized data to the client machine, making a TCP socket available to the application at both end points. At TCP transport layer, a congestion window is used at the sender 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, the TCP receiver informs the sender its current available space $awnd$, so that at the sender side, $cwnd \leq awnd$ condition is enforced at all times. At client application layer, a video player extracts data from a playout buffer, filled with packets delivered by the TCP receiver from its socket buffer. The playout buffer serves to smooth out variable data arrival rate.

A. Interaction between Video streaming and TCP

At the server side, HTTP server injects data into the TCP sender buffer according to $cwnd$ space availability. Hence, the injection rate of video data into the TCP socket is dictated by the congestion network condition, and thus different than the video variable encoding rate. Moreover, 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 data can be delivered without acknowledgements. Hence, for a fixed $cwnd$ size, from the sending of a first packet until the first acknowledgement arrives, a TCP session throughput is capped at $cwnd/RTT$. For each TCP congestion avoidance scheme, according to the TCP variant, 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 variants, 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 from TCP client socket by the video player into a playout buffer, before delivering to the video renderer. However, client 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.

IV. Transport Protocols

We now describe single/multipath transport protocols.

A. Multipath TCP

MPTCP is an IETF supported transport layer protocol which allows data transport over multiple TCP sessions [8]. The multipath nature of the transport session is hidden to upper layers via a single TCP socket use per application session. At the transport layer, however, MPTCP works with TCP variant sub-flows, each of which unaware of the multipath nature of the overall transport session. Connecting the application facing socket with transport sub-flow is a path scheduler, which extracts packets from the MPTCP socket exposed to applications, selects a sub-flow, and injects them into TCP sockets belonging to the selected sub-flow. MPTCP transport architecture is represented in Figure 1 (b).

The most widespread path scheduler (Linux implementation) selects the path with shortest round trip time (rtt) among paths with congestion window space for new packets. We refer to this path scheduler as default scheduler. In addition to this path scheduler, we include evaluation of a sticky scheduler [15], as follows. At the start of a new video streaming session, the path with smallest rtt is chosen, as per default scheduler. However, once a new path is selected (due to congestion of a previously selected path), the scheduler remains selecting the same path until it can no longer inject new packets. We call this path strategy as Greedy Sticky scheduler - GR-STY.

In addition, a MPTCP packet scheduler is supported, which adjusts the congestion window of each subflow according to some strategy. The packet scheduler may work in one of 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 several coupled mechanisms exist, we focus on Linked Increase Algorithm (LIA) [18] and Opportunistic Linked Increase Algorithm (OLIA) [11].

MPTCP supports the advertisement of IP interfaces available between two endpoints via specific TCP option signalling. As IP option signalling may be blocked by intermediate IP boxes such as firewalls, paths that cross service providers may require VPN protection. Morever, 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. TCP variants

TCP protocol variants can be classified into delay and loss based. Loss based TCP variants use packet loss as primary congestion indication signal, performing window regulation as $cwnd_k = f(cwnd_{k-1})$, hence being ack reception paced. Most f functions follow an Additive Increase Multiplicative Decrease (AIMD) strategy, with various increase and decrease parameters. TCP NewReno [3] and Cubic [19] are examples of AIMD strategies. Delay based TCP variants, 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 [20] and Capacity and Congestion Probing (CCP) [5] are examples of delay based protocols. Most TCP variants follow a slow start, congestion avoidance, fast retransmit and fast recovery phase framework. For TCP variants widely used, congestion avoidance is sharply different.

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 is adjusted as follows (1):

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

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. K parameter drives the cubic increase away from Wmax, whereas β tunes how quickly cwnd is reduced on packet loss. This adjustment strategy ensures that its $cwnd$ quickly recovers after a loss event.

Compound TCP Congestion Avoidance: Compound TCP is the TCP variant used in most deployed Wintel machines. This variant implements a hybrid loss/delay based congestion avoidance scheme, by adding a delay congestion window $dwnd$ to the congestion window of NewReno [20]. Compound TCP $cwnd$ adjustment is as follows (2):

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

where the delay component is computed as:

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

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	Cubic, Compound, LIA, OLIA
MPTCP schedulers	DFT, GR-STY

where parameter $diff$ is the estimated number of backlogged packets, γ is a threshold parameter which drives congestion detection sensitivity and α, β, η and K are parameters chosen as a tradeoff between responsiveness, smoothness and scalability. Compound TCP behavior is dominated by its loss based component, featuring a slow responsiveness to path bandwidth variations, which may cause playout buffer underflows.

Linked Increase Congestion Control: LIA [18] window adjustment couples the congestion control algorithms of different sub-flows by linking their congestion window increasing functions, while halving $cwnd$ window upon packet loss detection. LIA $cwnd$ adjustment scheme is as follows (4):

$$AckRec : cwnd_{k+1}^{i} = cwnd_{k}^{i} + min(\frac{\alpha B_{ack} Mss^{i}}{\sum_{0}^{n} cwnd^{p}}, \frac{B_{ack} Mss^{i}}{cwnd^{i}})$$

$$PktLoss : cwnd_{k+1}^{i} = \frac{cwnd_{k}^{i}}{2} \qquad (4)$$

where parameter α regulates the aggressiveness of the protocol, B_{ack} represents 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 (4) adopts $cwnd$ in bytes, rather than in packets (Maximum Segment Size - MSS), in contrast with other TCP variants equations, 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 with total increase no more than a single TCP Reno flow. The min operator in the increase adjustment equation guarantees that the increase is at most the same as if MPTCP was running on a single TCP Reno sub-flow. 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.

Opportunistic Linked Increase Congestion Control: OLIA [11] congestion window adjustment also couples the congestion control algorithms of different sub-flows, but with the increase based on the quality of the available paths. OLIA cwnd adjustment scheme is as follows (5):

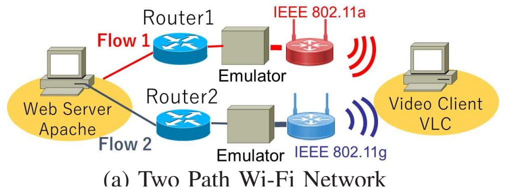
$$AckRec : cwnd_{k+1}^{i} = cwnd_{k}^{i} + \frac{\frac{cwnd^{i}}{(RTT^{i})^{2}}}{(\sum_{0}^{n} \frac{cwnd^{p}}{RTT^{p}})^{2}} + \frac{\alpha^{i}}{cwnd^{i}},$$

$$PktLoss : cwnd_{k+1}^{i} = \frac{cwnd_{k}^{i}}{2} \qquad (5)$$

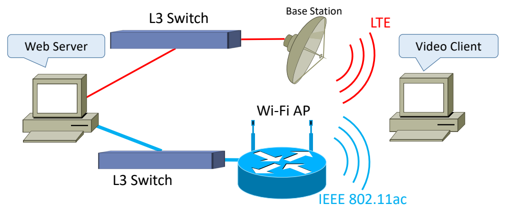
where α is a positive parameter for all paths. The idea is to tune $cwnd$ to an optimal congestion balancing point (Pareto optimal sense). In practical terms, each OLIA sub-flow increases $cwnd$ at a pace related to the ratio of each sub-flow RTT and the RTT of other subflows, still cutting $cwnd$ in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

V. STREAMING PERFORMANCE UNDER PATH HANDOFF

Figure 2 describes two network testbeds used for emulating network paths with WiFi and Cellular (LTE) wireless access links. In WiFi only testbed (a), an HTTP Apache video server



(a) Two Path Wi-Fi Network



(b) Cellular and Wi-Fi Network

Figure 2: Video Streaming Emulation Network

TABLE II: EXPERIMENTAL NETWORK SCENARIOS

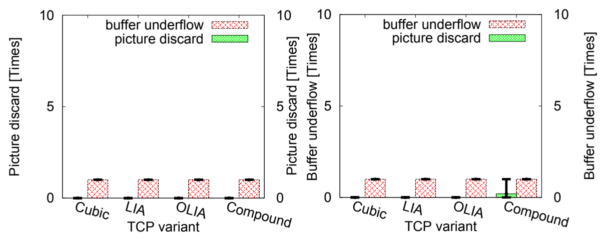
Scenario	Path properties (RTT, Bandwidth)
Limited BW Scenario Each path BW is close to video rate	Flow1) RTT 50 ms, BW 6 Mb/s Flow2) RTT 100 ms, BW 6 Mb/s
Large BW Scenario Each path BW is 3 times video rate	Flow1) RTT 50 ms, BW 18 Mb/s Flow2) RTT 100 ms, BW 18 Mb/s
Cellular Scenario (Interface BW speed)	Cellular) RTT 3.3ms, BW 24 Mb/s Wi-Fi) RTT 2.9ms, BW 433 Mb/s

is connected to two access routers, which are connected to link emulators, used to adjust path delays. A VLC client machine is connected to two Access Points, a 802.11a and 802.11g, on different bands (5GHz and 2.4GHz, respectively). In WiFi-Cellular testbed (b), an HTTP Apache video server is connected to two L3 switches, one of which directly connected to an 802.11ac router, and the other connected to an LTE base station via a cellular network card. The simple topologies and isolated traffic allow us to better understand the impact of differential delays, TCP variants, and path schedulers on streaming performance. Handoff is forced by cutting off WiFi primary path, simulating a break down of router to client communication.

Network settings and scenarios under study are described in Tables I and II, respectively. Video settings are typical of a video stream, with size short enough to run multiple streaming trials within a short amount of time. For WiFi only scenario, path bandwidth capacity is tuned to support a limited bandwidth and large bandwidth scenarios to stream a video playout rate of 5.24Mbps. TCP variants used are: Cubic, Compound, LIA and OLIA. Performance measures are:

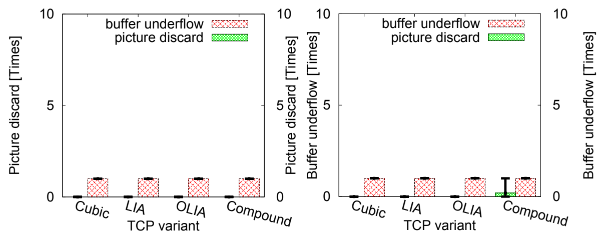
- **Picture discards:** number of frames discarded by the video decoder.
- **Buffer underflow:** number of buffer underflow events at video client buffer.
- **Sub-flow throughput:** the value of TCP throughput on each sub-flow.
- **Packet retransmissions:** number of packets retransmitted by TCP.

We organize our video streaming experimental results in three network scenarios (Table II): i) A WiFi-WiFi limited bandwidth scenario, with 6Mbps capacity on each path and differential delay; ii) A WiFi-WiFi large bandwidth scenario,



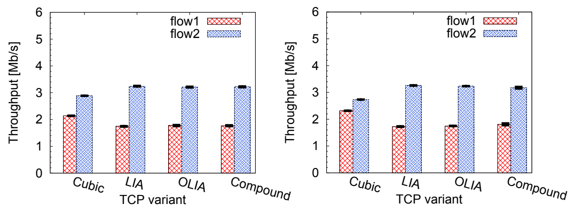
(a) Limited BW Scenario (b) Large BW Scenario

Figure 3: WiFi: no Handoff with DFT



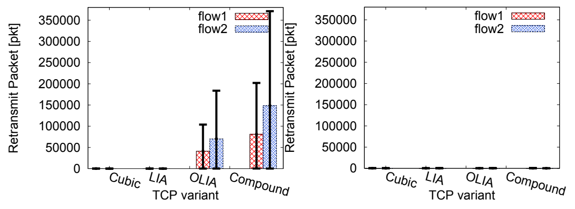
(a) Limited BW Scenario (b) Large BW Scenario

Figure 4: WiFi Handoff: video performance with DFT



(a) Limited BW Scenario (b) Large BW Scenario

Figure 5: WiFi Handoff: transport throughput with DFT



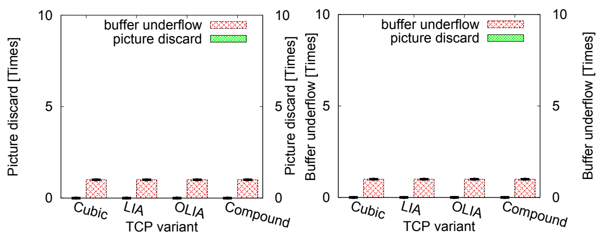
(a) Limited BW Scenario (b) Large BW Scenario

Figure 6: WiFi Handoff: transport retransmissions with DFT

with 18Mbps capacity on each path; iii) A WiFi-Cellular(LTE) scenario, with practically unlimited capacity on each path (path bandwidth is limited only by interfaces speed). Results are reported as average and min/max deviation bars.

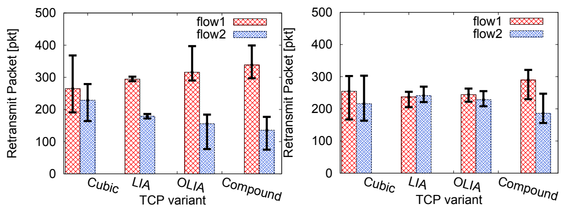
A. *WiFi Scenarios*

Figures 3 a and b report on video streaming and TCP performance of baseline scenario with no handoff, where flow 1 and 2 have 50, 100msec round trip times, respectively, and default packet scheduler. We see that picture discards and buffer underflows are as small as they can be, even when per flow bandwidth is limited (a). Figures 4 present same network scenario, but with WiFi-WiFi handoff. We see that for both limited and large bandwidth scenarios, video performance is not disturbed by handoffs. Figures 5 (a) and (b) report throughput of each flow, verifying that a larger throughput results on flow 2, with is the sole flow carrying traffic after handoff. Finally, for this handoff scenario, Figures 6 report



(a) Limited BW Scenario (b) Large BW Scenario

Figure 7: WiFi: No Handoff with GR-STY



(a) Limited BW Scenario (b) Large BW Scenario

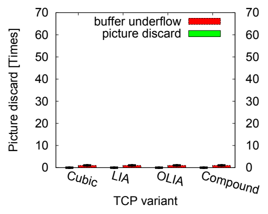
Figure 8: WiFi Handoff: transport retransmissions with GR-STY

on TCP layer packet retransmissions. Interestingly, in limited (tight) bandwidth scenario, significant retransmissions occur on both flow 1 and flow 2 for OLIA and Compound TCP variants. We notice that these two are the slowest variants to have their congestion window $cwnd$ recover from packet loss, as per respective equations of Section IV.

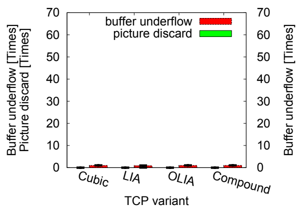
We have repeated handoff experiments using sticky scheduler instead of default scheduler, for comparison. Video performance results are similar to Figures 4, and hence are omitted for space's sake, as well as throughput results. However, transport retransmissions (Figures 8) show very little retransmissions on both flow 1 and flow 2 triggered by handoffs for all TCP variants (notice scale change of y-axis), including slow OLIA and Compound. From these and previous default scheduler retransmission results, we verified that large number of retransmissions occur prior to handoff, when both flow 1 and flow 2 are used, since the level of retransmissions is affected by the path scheduler used, with sticky scheduler alleviating retransmissions. Once handoff to flow 2 occurs, which occurs quickly due to both paths being available simultaneously, some extra retransmissions, no longer caused by the scheduler, also occur for OLIA and Compound TCP variants.

B. *Cellular Scenario*

Figures 9 a and b report on video streaming performance of WiFi - cellular network scenario with no handoff, under default and sticky path schedulers. We can verify perfect video streaming. In contrast, when handoffs from WiFi to cellular occur (Figures 10), buffer underflow and picture discards are significant for OLIA using default scheduler (a) , and LIA using sticky scheduler (b). Cubic and Compound TCP variants do not suffer video level performance degradation on under either path schedulers. In addition, Figures 11 confirm handoff from cellular link to WiFi link. Finally, Figures 12 show that most of retransmissions occur in the LTE path, for Compound and OLIA variants, again the least responsive variants to congestion window recovery.

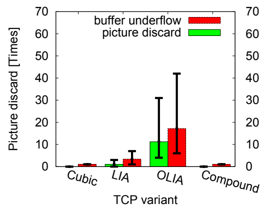


(a) DFT

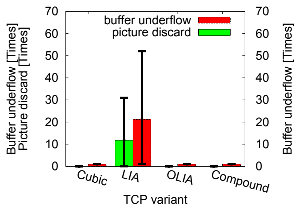


(b) GR-STY

Figure 9: WiFi-Cellular: No Handoff

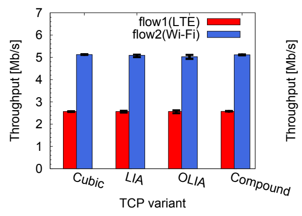


(a) DFT

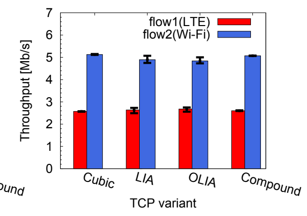


(b) GR-STY

Figure 10: WiFi-Cellular Handoff: video performance

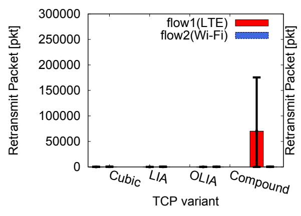


(a) DFT

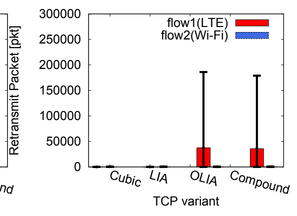


(b) GR-STY

Figure 11: WiFi-Cellular Handoff: transport throughput



(a) DFT



(b) GR-STY

Figure 12: WiFi-Cellular Handoff: transport retransmissions

Overall, the results show that video streaming over multiple paths may sustain handoffs between WiFi and cellular paths without significant performance degradation. In addition, sticky scheduler helps reduce retransmissions on slow to recover TCP variants such as OLIA and Compound.

VI. Conclusion and Future work

We have analyzed the impact of handoffs on video streaming performance over multiple paths. On a WiFi only scenario, we have shown that video streaming does not get affected by handoffs even on tight path bandwidth conditions. For WiFi-LTE cellular handoff, by using a VPN approach to overcome the issue of MPTCP signalling being dropped at intermediate nodes, we have shown video performance degradation for LIA and OLIA TCP variants. The path coupling of these TCP variants, where congestion window size depends on all active paths, slows down their recovery from packet losses during handoffs. We are currently investigating how coupled TCP variants may be made more robust to handoffs. We are also planning a handoff study on 5G cellular links.

Acknowledgments

Work supported by JSPS KAKENHI Grant # 16K00131.

References

[1] S. Afzal et al., "A Novel Scheduling Strategy for MMT-based Multipath Video Streaming," In Proceedings of IEEE Global Communications Conference - GLOBECOM, pp. 206-212, 2018.

[2] A. Aliyu et al., "Interference-Aware Multipath Video Streaming in Vehicular Environments," In IEEE Access Special Section on Towards Service-Centric Internet of Things (IoT): From Modeling to Practice, Volume 6, pp. 47610-47626, 2018.

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

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

[5] 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.

[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] A. Frommgen, J. Heuschkel and B. Koldehofe, "Multipath TCP Scheduling for Thin Streams: Active Probing and One-way Delay-awareness," IEEE Int. Conference on Communications (ICC), pp.1-7, May 2018.

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

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

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

[13] Matsufuji et al., "Multipath TCP Packet Schedulers for Streaming Video," IEEE Pacific Rim Conference on Communications, Computers and Signal Processing (PACRIM) , August 2017, pp. 1-6.

[14] Nagayama et al., "TCP State Driven MPTCP Packet Scheduling for Streaming Video," IARIA 10th International Conference on Evolving Internet, pp. 9-14, June 2018.

[15] Nagayama et al., "Path Switching Schedulers for MPTCP Streaming Video," IEEE Pacific Rim Conference on Communications, Computers and Signal Processing (PACRIM) , August 2019, pp. 1-6.

[16] R. K. P. Mok, E. W. W. Chan, and R. K. C. Chang, "Measuring the Quality of Experience of HTTP Video Streaming," Proceedings of IEEE International Symposium on Integrated Network Management, Dublin, Ireland, pp. 485-492, May 2011.

[17] Z. Lu, V. S. Somayazulu, and H. Moustafa, "Context Adaptive Cross-Layer TCP Optimization for Internet Video Streaming," In Proceedings of IEEE ICC 14, pp. 1723-1728, 2014.

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

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

[20] 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.

[21] F. Silva, D. Bogusevschi, and G-M. Muntean, "A MPTCP-based RTT-aware Packet Delivery Prioritization Algorithm in AR/VR Scenarios," In Proceedings of IEEE Intern. Wireless Communications & Mobile Computing Conference - IWCMCC 18, pp. 95-100, June 2018.

[22] H. Sinky, B. Hamdaoui, M. Guizani, "Proactive Multipath TCP for Seamless Handoff in Heterogeneous Wireless Access Networks," In Proceedings of IEEE Transactions on Wireless Communications, Volume 15, Issue 7, pp. 4754-4764, 2016.

[23] 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.