BBR Performance over Variable Delay Paths on Multipath TCP Video Streaming

Masayoshi Kondo*, Dirceu Cavendish**, Daiki Nobayashi**, Takeshi Ikenaga**
*Graduate School of Engineering, **Faculty of Engineering
Kyushu Institute of Technology
Fukuoka, Japan
e-mail: kondo.masayoshi146@mail.kyutech.jp {nova@ecs, ike@ecs}.kyutech.ac.jp

***Abstract*—Video streaming makes most of Internet traffic nowadays, being transported over Hypertext Transfer Protocol/Transmission Control Protocol (HTTP/TCP). Being the predominant transport protocol, TCP stack performance in transporting video streams has become paramount, specially with regard to MultiPath Transport Control Protocol (MPTCP) innovation and multiple client device interfaces currently available. An important component of MPTCP is the packet scheduler, which selects on a packet basis the transport path to inject each packet. In this paper, we provide an extensive analysis of the Bottleneck Bandwidth and Round-trip propagation time (BBR) TCP variant when transporting video streams over Long Term Evolution (LTE) and Wi-Fi access networks, comparing its performance to other available congestion control schemes and various path schedulers. We use network performance level, as well as video quality level metrics to characterize multiple path schedulers and the resulting network and application layers. We show that BBR video streaming performance degrades for challenging path delay variation scenarios.**

Keywords*—Video streaming; TCP congestion control; Multipath TCP; TCP BBR; Packet Scheduler.

I. INTRODUCTION

Reliable data transmission over the Internet relies on transport protocols to regulate data injection so as to control network congestion and avoid uncontrolled data losses along the communication path. In particular, TCP has become the defacto transport protocol of the Internet, supporting reliable data delivery for most applications. Regarding streaming applications, the most dominant type of application in data volume over the Internet, stream quality is related to two factors: the amount of data discarded at the client end point due to excessive transport delay/jitter and data rendering stalls due to lack of timely playout data. Transport delays and data starvation depend heavily on how TCP handles retransmissions upon packet losses during flow and congestion control.

Regarding multipath data delivery, the evolution of portable devices, in particular equipped with multiple high bandwidth interfaces, has motivated the development of the MultiPath Transport Control Protocol (MPTCP), allowing video streaming over multiple IP interfaces and diverse network paths to become reality. Multipath video streaming is attractive because it not only increases aggregated device downloading bandwidth capacity, but also improves transport session reliability during transient radio link impairments in handoff situations. An important function of multipath transport is the selection of a path among various active networking paths (sub-flows), which can be done on a packet by packet basis. However, a path packet scheduler should be designed so as to prevent head-of-line blocking across various networking paths, potentially with diverse loss and delay characteristics. Head-of-line blocking occurs when data already delivered at the receiver has to wait for additional packets that are blocked at another sub-flow, potentially causing incomplete or late frames to be discarded at the receiver, as well as stream rendering stalls. As TCP variants greatly impact streaming quality, we propose to analyze video performance vis-a-vis widely deployed TCP variants, with attention to Bottleneck Bandwidth and Round-trip propagation time (BBR) [1].

The paper is organized as follows. Related work is included in Section II. Section III describes video streaming transport over TCP, with focus on BBR and CUBIC TCP variants. Section VI characterizes video streaming performance over Wi-Fi and Long Term Evolution (LTE) paths via network emulation. We compare the application and network performance of BBR against CUBIC, using a default (shortest delay) path scheduler. Our goal is to uncover unfavorable network scenarios that may lead to the design of new path schedulers. Section VII summarizes our studies and addresses directions we are pursuing as follow up to this work.

II. RELATED WORK

Several multipath transport studies have appeared in the literature, mostly focusing on throughput performance of data transfers over mobile networks (see [2] and related work).

Meanwhile, the BBR TCP variant [1] has gained popularity, becoming Linux variant of choice. Only recent research work has focused on the performance evaluation of BBR in multipath transport. Austria et al. [3] carry an analysis of MPTCP in asymmetric latency subflows, focusing on throughput performance of data transfers. They compare favorably BBR against other TCP variants, such as BIC and CUBIC, on latency asymmetric subpath network scenarios. Our contribution is similar to theirs, but focusing on video streaming applications. In Mahmud et al. [4], a coupling of BBR with MPTCP packet scheduler is proposed with the goal of maintaining high throughput performance while achieving fairness among different subflows. They use emulation of different subflow scenarios to compare their coupled MPTCP throughput and fairness performance against other TCP variants, including Linked Increase Algorithm (LIA) [5], Opportunistic Linked Increase Algorithm (OLIA) [6], Balanced Linked Adaptation algorithm (BALIA) [7]. In contrast, this work evaluates the video streaming performance of BBR and other TCP variants, using various schedulers.

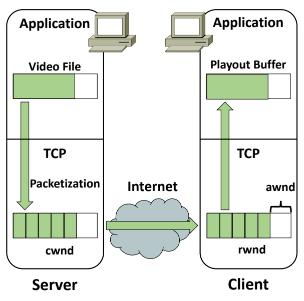
Little research work has focused on video streaming performance over multiple paths. In Matsufuji et al. [8], we evaluate the performance of several TCP variants and path schedulers in transporting video streams over multipath, quantifying frame discards and play stalls. Morawski et al. [10] conduct Linux based experiments of multipath video streaming over Digital Subscriber Line (DSL) path scenarios using LIA, and OLIA, as well as Reno, CUBIC, and BBR TCP variants. They show head-of-line blocking as a major concern. Unfortunately, they do not provide application level performance measures, to evaluate video quality impact. Similarly, Amend et al. [11] evaluate throughput of multipath video streaming DSL multipath scenarios, without providing video level performance measures. Although they also propose a cost optimized scheduler, the lack of video quality performance measures limits conclusions about impact of such scheduler to video quality. Along the same lines, Imaduddin et al. [12] provide a performance evaluation of MPTCP using CUBIC and Vegas TCP variants, as well as minimum Round Trip Time (RTT), round-robin and coupled BALIA schedulers. Focusing on throughput performance, they conclude CUBIC to deliver best performance, regardless of the scheduler. Finally, Xing et al. [13] propose a new MPTCP scheduler which they show via network experiments to lower the number of out-of-order packets. The scheduler estimates receiver arrival times, and sends redundant packets to cope with estimation errors. Video streaming is simulated via $iperf3$, and no application layer performance measures are used. Among all these works, our line of research has focused on application level performance measures in addition to network layer performance indicators such as throughput. In our previous works, we have also introduced multipath path scheduling generic principles, which can be applied in the design of various path schedulers to specifically improve video stream quality. Using these principles, we have introduced in [8] MPTCP path schedulers based on dynamically varying path characteristics, such as congestion window space and estimated path throughput. In addition, in Nagayama et al. [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 states. Finally, in Nagayama et al. [9], we have introduced a novel concept of sticky scheduling, where once a path switch is executed, the scheduler stays with the new path until the path bandwidth resources become exhausted. In this work, we evaluate multipath video streaming using only the default minimum RTT path scheduler, in combination with popular BBR TCP [1] over realistic Wi-Fi/Cellular multipath scenarios, focusing on video quality at the application layer. We seek to determine whether BBR delivers high performance in combination with the default path scheduler over challenging variable delay network scenarios. BBR performance in MPTCP transport is a novelty. Han et al. [15] have recently introduced one such study, where BBR is evaluated in combination with a new adaptive packet scheduling scheme (adaptive redundant + predictive). Multiple copies of a packet are injected in paths of low quality, for reliability improvement. The scheme also predicts packet delivery on paths in order to keep in order delivery, mitigating head-of-line blocking. Their evaluation, however, is limited to throughput and download time performance metrics of files. In our previous evaluation of BBR [16], we have provided an evaluation of BBR when transporting multipath video streams over lossy paths, which showed superior performance vis-a-vis popular TCP variants. In this study, we further evaluate BBR on various path delay characteristics. We show that performance degradation is indeed experienced in some scenarios.

III. Video Streaming Over MPTCP

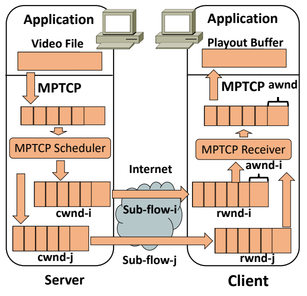
Video streaming over Hypertext Transfer Protocol/Transmission Control Protocol (HTTP/TCP) originates at a HTTP server storing video content, where video files can be streamed upon HTTP requests over the Internet to video clients. At the transport layer, a TCP variant provides reliable transport of video data over IP packets between the server and client end points (Figure 1). Upon an HTTP video request, a TCP sender is instantiated to transmit packetized data to the client machine, connected to the application via a TCP socket. At the TCP transport layer, a congestion window is used at the sender to control the amount of data injected into the network. The size of the congestion window ($cwnd$) is adjusted dynamically, according to the level of congestion experienced at the network path, as well as space available for data storage ($awnd$) at the TCP client receiver buffer. the congestion window space at the sender is freed only when data packets are acknowledged by the receiver. Lost packets are retransmitted by the TCP layer to ensure reliable data delivery. At the client end, in addition to acknowledging arriving packets, the TCP receiver informs the TCP sender about its current receiver available space, so that the $cwnd \leq awnd$ condition is enforced by the sender at all times to prevent receiver buffer overflow. At the client application layer, a video player extracts data from a playout buffer, which draws packets delivered by the TCP receiver from receiver TCP socket buffer. The playout buffer hence serves to smooth out variable network throughput.

A. MPTCP

MPTCP is an Internet Engineering Task Force (IETF) extension of TCP transport layer protocol to support data transport over multiple concurrent TCP sessions [17]. The network multipath transmission of the transport session is hidden from the application layer by a legacy TCP socket exposed per application session. At the transport layer, however, MPTCP coordinates concurrent TCP sessions on various sub-flows, each of which in itself is unaware of the multipath nature of the application session. In order to accomplish multipath



(a) TCP



(b) MPTCP

Figure 1. Video Streaming over TCP/MPTCP

transport, a path scheduler connects the application socket with transport sub-flows, extracting packets from the application facing MPTCP socket, selecting a sub-flow for transmission, and injecting packets into the selected sub-flow. the MPTCP transport architecture is depicted in Figure 1 (b).

The first and most used path scheduler, called default scheduler, selects the path with the shortest RTT among paths with currently available congestion window space for new packets. Other path schedulers have appeared recently. These path schedulers can operate in two different modes: uncoupled, and coupled. In uncoupled mode, each sub-flow congestion window $cwnd$ is adjusted independently of other sub-flows. On the other hand, in coupled mode, the MPTCP scheduler couples the congestion control of the sub-flows, by adjusting the congestion window $cwnd_k$ of a sub-flow k according to the current state and parameters of all available sub-flows. Although many coupling mechanisms exist, we focus on the performance study of the Bottleneck Bandwidth and round trip [1] TCP variant over the shortest path RTT scheduler.

Regardless of the path scheduler used, the IETF MPTCP protocol supports the advertisement of multiple IP interfaces available between two endpoints via specific TCP option signalling. IP interfaces may be of diverse nature (e.g., Wi-Fi, LTE). A common signalling issue is caused by intermediate IP boxes, such as firewalls, blocking IP options. Paths that cross service providers with such boxes may require Virtual Private Network (VPN) protection so as to preserve IP interface advertising between endpoints. In addition, multipath transport requires MPTCP stack at both endpoints for the establishment and usage of multiple paths.

IV. TCP VARIANTS

TCP protocol nowadays has branched into different variants, implementing different congestion window adjustment schemes. TCP protocol variants can be classified into delay- and loss-based congestion control schemes. Loss-based TCP variants use packet loss as primary congestion indication signal, typically performing congestion window regulation as $cwnd_k = f(cwnd_{k-1})$, which is ack reception paced. Most f functions follow an Additive Increase Multiplicative Decrease (AIMD) window adjustment scheme, with various increase and decrease parameters. AIMD strategy relies on a cautious window increase (additive) when no congestion is detected, and fast window decrease (multiplicative) as soon as congestion is detected. TCP NewReno [18] and CUBIC [19] are examples of AIMD strategies. In contrast, delay based TCP variants 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) [21] are examples of delay based congestion control variants. Delay based congestion control does not suffer from packet loss undue window reduction due to random, not congestion, packet losses, as experienced in wireless links. Regardless of the congestion control scheme, TCP variants follow a phase framework, with an initial slow start, followed by congestion avoidance, with occasional fast retransmit, and fast recovery phases. BBR congestion control may be considered delay based, since BBR measures the bandwidth and RTT of the bottleneck which a flow goes through [1]. Based on such measurements, BBR adjusts the sending rate to make the best use of the bottleneck bandwidth without dropping its rate during wireless link random losses.

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.

BBR TCP Congestion Avoidance: BBR is a bandwidth delay product based TCP that has achieved widespread usage as one of available TCP variants in the Linux operating system. BBR uses measurements of a connection delivery rate and RTT to build a model that controls how fast data may be sent and the maximum amount of unacknowledged data in the pipe. Delivery rate is measured by keeping track of the number of acknowledged packets within a defined time frame. In addition, BBR uses a probing mechanism to determine the maximum delivery rate within multiple intervals.

More specifically, BBR regulates the number of in-flight packets to match the bandwidth delay product of the connection, or $BDP = BtlBw \times RTprop$, where $BtlBw$ is the bottleneck bandwidth of the connection, and RTprop its propagation time, estimated as half of the connection RTT. These quantities are tracked during the lifetime of the connection, as per equations below (2):

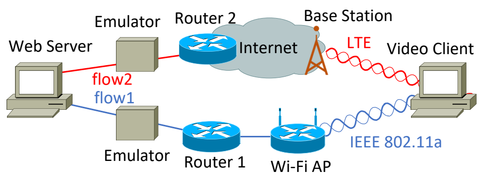


Figure 2. Video Streaming Emulation Network

TABLE I. EXPERIMENTAL NETWORK SETTINGS

Element	Value
Video size	113 MBytes
Video rate	5.24 Mb/s
Playout time	3 mins
Video Codec	H264 MPEG-4 AVC
MPTCP variants	BBR, CUBIC
MPTCP schedulers	Low RTT First (Default)

$$\begin{aligned} RTT_t &= RTprop_t + \eta_t \\ R\hat{T}prop &= RTprop + min(\eta_t) \\ &= min(RTT_t) \forall t \in [T - W_R, T] \\ Btl\hat{B}w &= max(deliveryRate_t) \forall t \in [t - W_B, T] \end{aligned} \quad (2)$$

where η_t represents the noise of the queues along the path, W_R a running time window, of tens of seconds, and W_B a larger time window, of tens of RTTs. This adjustment strategy seeks to tune its *cwnd* to a number of packets equivalent to the connection bandwidth delay product.

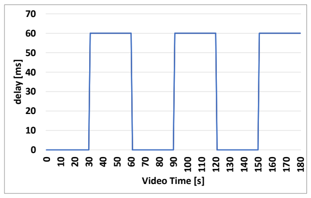
V. MPTCP WITH TCP BBR

A MPTCP scheduler selects a 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 round trip packet delay, hereafter called LRF. If a short and non-congested path exists between the end points, it becomes the preferred path for data transport. Non-congested path is defined as a path for which its congestion window (*cwnd*) has available space among packets yet to be acknowledged by the receiver. Hence, a congested path will have no space for more unacknowledged packets to be injected.

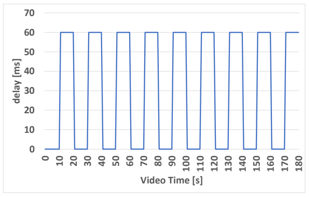
As BBR sizes its *cwnd* according to path bandwidth delay product, paths with large delays and high bandwidth result in large *cwnd*. Even though the scheduler may favor a path with smaller bandwidth delay product, as it looks at the path RTT only, if such path has low bandwidth availability BBR will size its *cwnd* to a small value as compared to high bandwidth paths, effectively blocking low bandwidth paths from scheduling selection, and forcing a high bandwidth delay product path to be used.

VI. VIDEO STREAMING PERFORMANCE OVER WI-FI/LTE

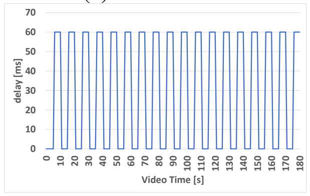
Figure 2 describes the network testbed used for emulating network paths with Wi-Fi and LTE wireless access links. An HTTP Apache video server is connected to two L3 switches, one of which directly connected to an 802.11a router, and the other connected to an LTE base station via a cellular network card via emulator boxes. Since the bandwidth of IEEE 802.11a is sufficiently large for the bit rate of video, we have adopted 802.11a as the wireless LAN interface. In this paper, the emulator boxes are used to vary each path RTT. No packet loss is injected. The simple topology and isolated traffic allow us to better understand the impact of differential delays on TCP variant's performance.



(a) Scenario A



(b) Scenario B



(b) Scenario C

Figure 3. Wi-Fi Delay Dynamic Change Scenario

TABLE II. EXPERIMENTAL NETWORK SCENARIOS

Scenario	Emulator (BW, Packets Loss)
A - Slow Wi-Fi delay cycle Scenario A delay pattern	LTE: BW 3Mbps, Loss 0% Wi-Fi: BW 3Mbps, Loss 0% DCycle 60secs
B - Baseline Wi-Fi delay cycle Scenario B delay pattern	LTE: BW 3Mbps, Loss 0% Wi-Fi: BW 3Mbps, Loss 0% DCycle 20secs
C - Fast Wi-Fi delay cycle Scenario C delay pattern	LTE: BW 3Mbps, Loss 0% Wi-Fi: BW 3Mbps, Loss 0% DCycle 10secs

Network settings and scenarios under study are described in Tables I and II, respectively. Video settings are typical of a video stream, with video playout rate of 5.24 Mb/s, and size short enough to run multiple streaming trials within a short amount of time. Three Wi-Fi packet delay pattern scenarios are used (Figure 3). Scenario A represents streaming sessions with steady Wi-Fi delays, with ON 60 msec cycles of 60 seconds. Scenario B represents a baseline Wi-Fi delay scenario, where 60 msec delay happen at 20 sec cycles. Scenario C represents a highly variable delay scenario, where 60 msec delay is experienced at a faster 10 second cycles. Emulator boxes are tuned to generate various multiple path network conditions, and have been selected as per Table II to represent LTE/Wi-Fi streaming situations at home. TCP variants used are: CUBIC and BBR. 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:** TCP throughput on each sub-flow.
- **Sub-flow cwnd:** TCP cwnd value on each sub-flow.

We organize our video streaming experimental results in network scenarios summarized in Table II: A- A Wi-Fi-Cellular (LTE) scenario A high frequency delay cycle; B- A Wi-Fi-Cellular scenario B, where a slightly larger Wi-Fi delay cycles is assumed as baseline; C- A Wi-Fi-Cellular with scenario C high cycle delay pattern.

A. Slow delay on Scenario A delay pattern

Scenario A delay represents slow varying path delays, emulating a transition between stable Wi-Fi low load connections. Figures 4(a) and (b) report on video streaming buffer underflow and picture discard performance when Wi-Fi delay is slow varying about 60 ms. Video performance is excellent for both BBR and CUBIC TCP variants (single buffer underflow event and zero picture discards). Figures 5(a) and (b) report of Wi-Fi *cwnd* dynamics of BBR and CUBIC TCP variants. We can see that BBR enforces a much reduced Wi-Fi *cwnd* than CUBIC, still delivering excellent video performance. BBR *cwnd* size tracks nicely delay cycles, differently from CUBIC, which, being a loss based variant, is insensitive to delay variations. Moreover, Figures 5(c) and (d) show similar levels of LTE (cellular) and Wi-Fi path throughput for both TCP variants, showing an equal share of the two paths available, despite Wi-Fi path delay variability.

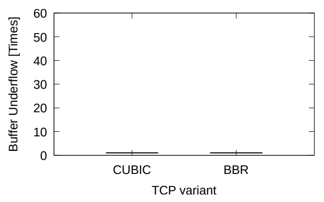
B. Baseline delay on Scenario B delay pattern

Scenario B delay represents medium varying path delays, taken as a baseline Wi-Fi scenario. Figures 6(a) and (b) report on video streaming buffer underflow and picture discard performance when Wi-Fi delay is varying on 20 seconds cycles of 60 ms delay. Video performance degrades for both TCP variants, with BBR delivering less buffer underflows and more picture discards than CUBIC. Figures 7(a) and (b) report of Wi-Fi *cwnd* dynamics of BBR and CUBIC TCP variants. Again, BBR enforces a much reduced Wi-Fi *cwnd* than CUBIC, still tracking delay cycles nicely. Moreover, Figures 7(c) and (d) show BBR delivering a reduced level of Wi-Fi throughput than CUBIC, using more LTE bandwidth than CUBIC.

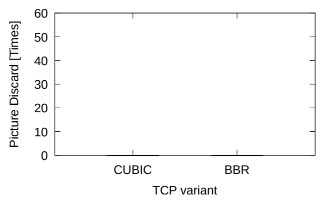
C. Small delay on Scenario C delay pattern

Scenario C delay represents highly varying path delay Wi-Fi scenario. Figures 8(a) and (b) report on video streaming buffer underflow and picture discard performance when Wi-Fi delay is varying on 10 seconds cycles of 60 ms delay. Video performance degrades significantly for BBR, whereas CUBIC delivers video performance comparable with previous scenario. Figures 9(a) and (b) report of Wi-Fi *cwnd* dynamics of BBR and CUBIC TCP variants, and helps explain BBR performance degradation. BBR is no longer able to track delay cycles as before, remaining ”stuck” at a small *cwnd* of 15 packets. Moreover, Figures 9(c) and (d) show BBR delivering a much reduced level of Wi-Fi throughput than CUBIC, and not being able to compensate enough with more LTE bandwidth than CUBIC. The overall video connection, therefore, gets starved when served by BBR.

Finally, Figure 10 reports on the number of path switches of both TCP variants on the three scenarios investigated. We see that the number of path switches when using BBR is as much as four times larger than CUBIC. This is because there is a stress between the delay sensitivity of BBR vs the default scheduler minimum delay path selection. That is, a

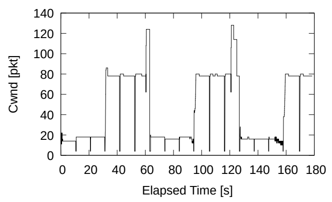


(a) Buffer Underflow

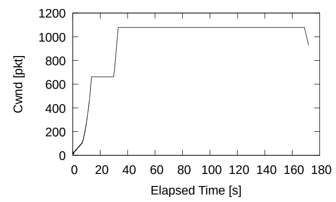


(b) Picture Discard

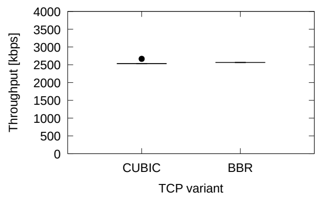
Figure 4. Scenario A - Video Performance



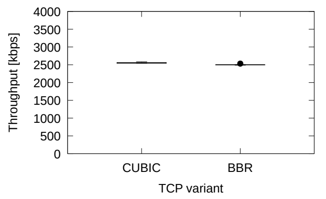
(a) BBR : cwnd, Wi-Fi



(b) CUBIC : cwnd, Wi-Fi



(c) Throughput, LTE



(d) Throughput, Wi-Fi

Figure 5. Scenario A - Transmission Performance

delay variation causes BBR to shrink its *cwnd*, effectively blocking that path quickly, pushing the scheduler to switch to LTE path. However, as soon as there is any room in the Wi-Fi path, the scheduler switches paths back to Wi-Fi. This indicates that a scheduler that follows BBR bandwidth and delay estimation may be able to work more harmoniously with BBR variant.

VII. Conclusion

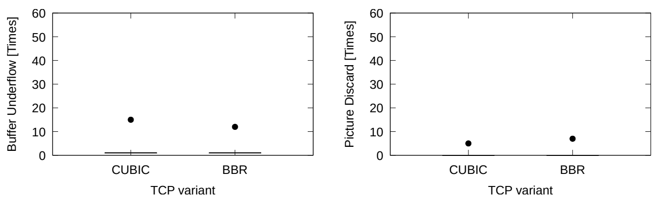
We have studied BBR transport performance of video streaming on multipath cellular/Wi-Fi mixed scenarios. We have shown that on rapidly varying path delay scenario, BBR TCP variant delivers a degraded video streaming performance. Under this fast delay variation, BBR remains at a shrunk congestion window situation that effectively reduces considerably the path throughput. These early results seem to point to an opportunity of designing a path scheduler that is better tuned to delay and bandwidth delay product based TCP variants such as BBR. For instance, path scheduler could select the path with larger bandwidth delay product, rather than lower delay of default scheduler. We are currently designing one such scheduler, where BBR estimators are used to drive path selection.

Acknowledgments

Work supported by JSPS KAKENHI Grant #21H03430, and National Institute of Information and Communication Technology (NICT).

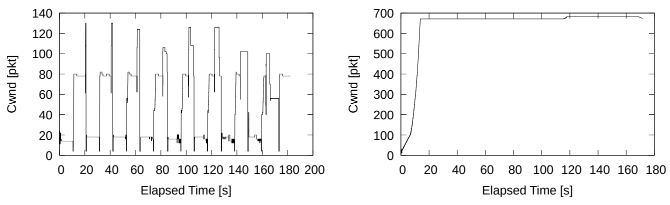
References

[1] N. Cardwell, Y. Cheng, I. Swett, and V. Jacobson, “BBR Congestion Control,” *IETF draft-cardwell-iccrg-bbr-congestion-control-01*, November 2021.
[2] M. R. Palash et al., “MPWiFi: Synergizing MPTCP Based Simultaneous Multipath Access and WiFi Network Performance,” IEEE Transactions on Mobile Computing, vol. 19, no. 1, pp. 142-158, Jan. 2020.

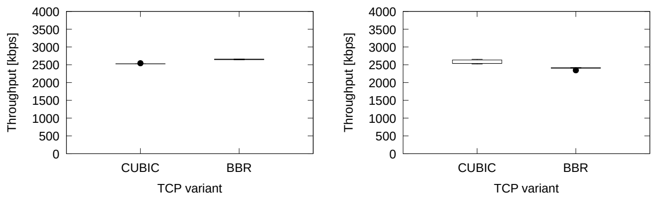


(a) Buffer Underflow (b) Picture Discard

Figure 6. Scenario B - Video Performance

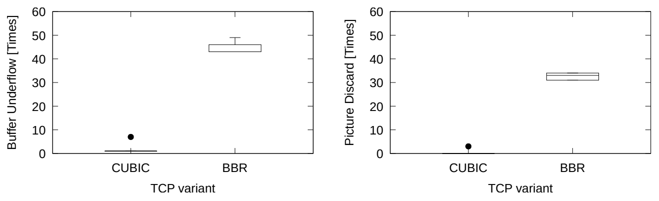


(a) BBR : cwnd, Wi-Fi (b) CUBIC : cwnd, Wi-Fi



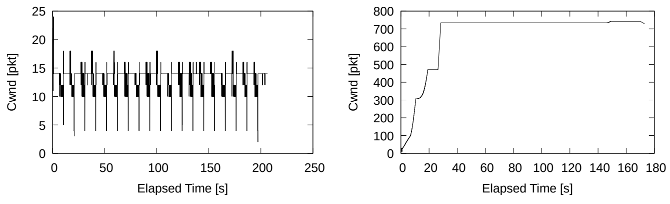
(c) Throughput, LTE (d) Throughput, Wi-Fi

Figure 7. Scenario B - Transmission Performance

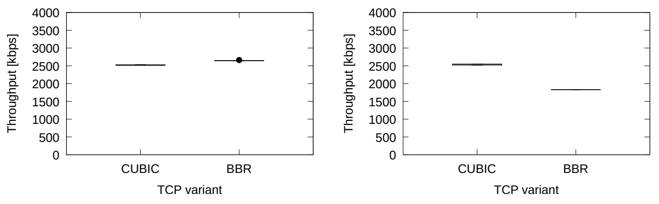


(a) Buffer Underflow (b) Picture Discard

Figure 8. Scenario C - Video Performance



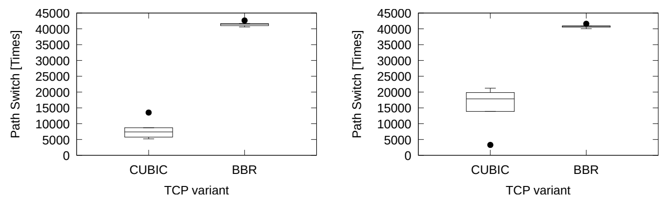
(a) BBR : cwnd, Wi-Fi (b) CUBIC : cwnd, Wi-Fi



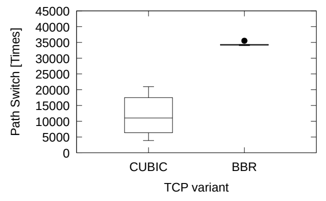
(c) Throughput, LTE (d) Throughput, Wi-Fi

Figure 9. Scenario C - Transmission Performance

[3] P. Austria et al., "BBR Congestion Control Analysis with Multipath TCP (MPTCP) and Asymmetrical Latency Subflow," In Proc. of IEEE 12th Annual Computing and Communication Workshop - CCWC, pp. 1065-1069, January 2022.
[4] I. Mahmud et al., "Coupled Multipath BBR (C-MPBBR): A Efficient Congestion Control Algorithm For Multipath TCP," IEEE Access, vol. 8, pp. 165497-165511, Sept. 2020.
[5] C. Raiciu, M. Handly, and D. Wischik, "Coupled Congestion Control



(a) Scenario A : Path Switch (b) Scenario B : Path Switch



(c) Scenario C : Path Switch

Figure 10. All Scenarios - Path Switch

for Multipath Transport Protocols," IETF RFC 6356, 2011.
[6] 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.
[7] A. Walid et al., "Balanced Linked Adaptation Congestion Control Algorithm for MPTCP," IETF draft-walid-mptcp-congestion-control, 2014.
[8] R. 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.
[9] S. 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.
[10] M. Morawski and P. Ignaciuk, "A Price to Pay for Increased Throughput in MPTCP Transmission of Video Streams," In Proc. of 24th Intern. Conference on System Theory, Control and Computing - ICSTCC, pp. 673-678, October 2020.
[11] M. Amend et al., "Cost optimized multipath scheduling in 5G for Video-on-Demand traffic," In Proc. of IEEE Wireless Communications and Networking Conference - WCNC 21, March 2021.
[12] M. F. Imaduddin et al., "Multipath TCP Scheduling Performance Analysis and Congestion Control on Video Streaming on the MPTCP Network," In Proc. of Intern. Conference on Software Engineering & Computer Systems and 4th Intern. Conference on Computational Science and Information Management - ICSECS-ICOCSIM, pp. 562-567, August 2021.
[13] Y. Xing et al., "A Low-Latency MPTCP Scheduler for Live Video Streaming in Mobile Networks," IEEE Transactions on Wireless Communications, vol. 20, no. 11, pp. 7230-7242, Nov. 2021.
[14] S. 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] J. Han et al., "Leveraging Coupled BBR and Adaptive Packet Scheduling to Boost MPTCP," IEEE Transactions on Wireless Communications, vol. 20, no. 11, pp. 7555-7567, November 2021.
[16] M. Kondo et al., "Evaluation of MPTCP with BBR Performance on Wi-Fi/Cellular networks for Video Streaming," IARIA 14th International Conference on Evolving Internet, pp. 6-11, May 2022.
[17] A. Ford et al., "Architectural Guidelines for Multipath TCP Development," IETF RFC 6182, 2011.
[18] M. Allman, V. Paxson, and W. Stevens, "TCP Congestion Control," IETF RFC 2581, April 1999.
[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] 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.