An Evaluation of Multipath TCP in Lossy Environment

Kien Nguyen[*], Mirza Golam Kibria[†], Kentaro Ishizu[†], Fumihide Kojima[†], Hiroo Sekiya[*]
[*]Graduate School of Engineering, Chiba University
1-33, Yayoi-cho, Inage-ku, Chiba-shi, Chiba, 263-8522 Japan
[†]Wireless Systems Laboratory, National Institute of Information and Communications Technology
3-4 Hikarinooka, Yokosuka, Kanagawa, 239-0847 Japan
Email: nguyen@chiba-u.jp, mirza.kibria@uni.lu, {ishidu, f-kojima}@nict.go.jp, sekiya@faculty.chiba-u.jp

***Abstract*—Multipath TCP (MPTCP) has been increasingly adopted in the current and next generation of mobile wireless networks. MPTCP can exploit multiple wireless links for an application; hence it enhances not only the reliability but also the performance of TCP in the same scenario. However, most of the state-of-the-art congestion controls for MPTCP share the same behavior of TCP when a lost packet happens on a path. Therefore, MPTCP also incurs performance degradation when the number of lost packets increases (i.e., the well-known problem of TCP). Comparing to TCP, the deterioration in MPTCP maybe even worse, especially when the loss simultaneously happens on multiple paths. Since the lossy paths likely occur in wireless networks, the improvement of MPTCP in such environment is necessary. In this work, we explore the feasibility of applying a novel approach of handling loss in TCP congestion control to MPTCP. Specifically, we investigate and extend the bottleneck bandwidth round trip time (BBR) algorithm, which reacts to the loss signal in TCP following an estimation model, for MPTCP. We integrate BBR to MPTCP in a new implementation, namely *mpbbr*. We then evaluation *mpbbr* under the lossy networks in comparison to the MPTCP with the *balia* congestion control. The evaluation results show that *mpbbr* well handles against the lossy conditions as well as outperforms *balia* in the investigated scenarios.**

Keywords: *MPTCP, balia, bbr, loss-handling, evaluation*

I. Introduction

In recent years, the advances in wireless technologies and the popularity of mobile devices have shifted the way of accessing the Internet from wired networks to wireless ones. We usually use Internet services on a mobile device, which can reach the Internet core via different wireless links (e.g., Wi-Fi and LTE in 4G). In the upcoming generation of mobile wireless networks (i.e., 5G), the device will have additional radios (e.g., 5G New Radio, millimeter wave, etc.), which promisingly bring the new level of quality of experience for mobile Internet users. Therefore, it is necessary to harness the resource of multiple wireless networks for applications. Multipath TCP (MPTCP) has emerged as a promising technology since it supports the bandwidth aggregation of existing wireless links, as well as, soft handover between them. MPTCP not only concurrently exploits multiple wireless connections but also requires zero modification in the application layer. MPTCP has been standardized by IETF [1] with an increasing number of applications and use cases [2]. MPTCP has been actively developed and adopted on popular platforms such as Linux, Google Android [3] and Apple iOS [4].

Mirza Golam Kibria is now with SIGCOM Research Group, SnT, University of Luxembourg.

MPTCP uses the standard socket to interface with the application layer. It separates an application's byte stream into different TCP flows (i.e., MPTCP subflows). The packets on each subflow is divided and scheduled to send via an end-to-end path (between a pair of IP addresses). The divided packets are finally resequenced at one end of communication. To handle the sequencing issue, a subflow contains additional data sequence besides the one in each subflow. In general cases, MPTCP handles sequencing/resequencing well enough to realize the aggregation of throughput on several paths. MPTCP has been proven to be efficient in various wireless networks (e.g., 3G/Wi-Fi [5], Wi-Fi/Wi-Fi [6], virtual Wi-Fi/virtual Wi-Fi [7], or software defined wireless networks [8], etc.). However, when a network path experiences a large enough number of lost packets, the retransmissions may be harmful to the overall performance of MPTCP. Since it is hard to avoid random loss in the wireless environment, MPTCP should be enhanced in such situations.

In MPTCP, the congestion control module is in charge of reaction against loss. However, all the major MPTCP congestion controls (i.e., *lia* [9], *olia* [10], *wvegas* [11], *balia* [12]) use the unmodified behavior of TCP in such cases. Therefore, MPTCP still incurs the problem of performance degradation under lossy conditions as TCP does. The degradation may become worse in MPTCP since the resequence at an end point needs data packets on multiple paths. This work investigates the feasible adoption of a recent advanced development of TCP congestion control for MPTCP aiming to enhance the MPTCP efficiency in lossy environments. Specifically, we opt the model based approach of the bottleneck bandwidth and round-trip time (BBR) congestion control proposed by Google [13] to MPTCP. In the model of BBR, the transmission rate of a TCP sender is an estimation of bottleneck bandwidth and round-trip time. Different to the traditional loss-based or delay-based TCP congestion control, BBR can avoid the large bottleneck buffer that causing bufferbloat by sending a suitable amount of data to the network. More importantly, BRR interprets the loss signal differently. BRR does not directly behave according to the packet lost. Instead it relies on the estimation to reduce the sending rate. We first integrate BRR to the MPTCP kernel in a new implementation namely *mpbbr* (MPTCP with BBR). We then evaluate the *mpbbr* performance under different loss conditions in comparison to *balia*. The results show that *mpbbr* handles random loss much better than *balia*. Moreover, *mpbbr* can provides a good aggregated throughput even when the serious loss concurrently happens on multiple paths.

The remainder of this paper is organized as follows. Section II presents the background of MPTCP, congestion control and BBR. Section III shows the evaluation results. The related work is introduced in Section IV. Finally, Section V indicates the conclusion and future works.

II. Multipath TCP and BBR

This section briefly introduces an overview of MPTCP as well as its congestion controls. We also describe the basic operation of bottleneck bandwidth round trip time (BBR).

A. Multipath TCP

An MPTCP connection begins with a three-way handshaking (e.g., the red SYN, SYN/ACK, ACK packets in Fig. 1), which is similar to the one in TCP. Different to TCP, MPTCP's packets have an additionally field of the MP_CAPABLE option, that is for checking the MPTCP capability. Those packets also have extra flags, which are necessary for the usage of checksum and cryptographic. If the SYN arriving at the receiver does not contain the MP_CAPABLE option, the MPTCP connection is not initialized. Instead, the following progress is as same as in TCP. On the other hand, the initialization of MPTCP connection is completed. Each MPTCP connection is uniquely identified by a pair of keys between a client and a server, which use for the verification of later subflow. To add a subflow to an MPTCP connection, a new handshake with MP_JOIN option is utilized (i.e., in the lower part of Fig. 1). After successful establishments, the data is concurrently transferred through two paths. MPTCP can schedule to use whole set or subset of available paths for data transmission. Therefore, MPTCP enables the advanced features of link aggregation and soft handover. The reacts of subflows against network conditions are controlled by congestion control algorithms.

MPTCP has been designed to support two types of congestion control algorithms, which are uncoupled and coupled. The former one indicates the independent between the congestion control for each subflow. Each subflow operates exactly as a TCP flow. Therefore, it can use any available TCP congestion control such as *reno, cubic, westwood, vegas*, etc. The later one is proposed with the consideration of other subflow transmission (e.g., fairness between subflows) [14]. It aims to couple the congestion windows of all subflows, which belong to an MPTCP connection, with a resource pooling principle. The coupled congestion controls let MPTCP change the sending rate of each subflow while improve the bottleneck fairness. There have been several coupled congestion controls, *lia* (Linked Increases Algorithm), *olia* (Opportunistic LIA), *balia*

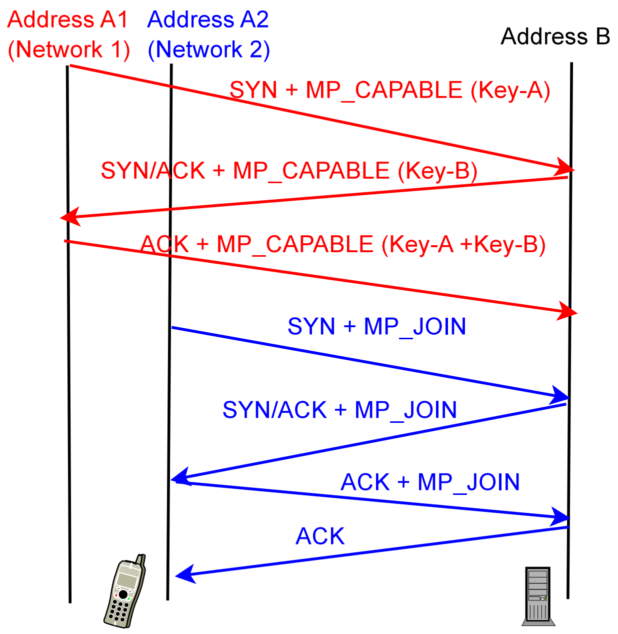


Fig. 1: Brief Introduction of MPTCP Connection

(Balanced LIA) and *wvegas* (Weighted Vegas). Those coupled congestion controls only affects the increasing phase of the congestion avoidance state, specifying how the congestion window inflates upon receiving an acknowledgement. Other phases are mostly as same as the ones in TCP. Among the above congestion control, *balia* is the recent proposal, which outperforms the others in various aspects. Therefore, we select *balia* as the baseline in this work.

B. Balia and Bottleneck Bandwidth and Round-trip propagation time (BBR)

As previously mentioned, the balance linked increased algorithm (*balia*) [12] has been proven to have better performance than the others. The authors in [12] not only provide mathematical proofs but also the real implementation of *balia*. The behavior of *balia*'s sender is determined following the ACK and loss signals. When it receives an ACK on the subflow i, the window size w_i is as follows:

$$w_i \longleftarrow w_i + \frac{w_i/(RTT_i)^2}{\sum_{p \in R}(w_p/RTT_p)^2} * \frac{1+\alpha_i}{2} * \frac{4+\alpha_i}{5}$$

When a loss is recognized on the subflow i, the window size is calculated as follows:

$$w_i \longleftarrow w_i - \frac{w_i}{2} * min\{\alpha_i, 1.5\}$$

in which $\alpha_i = \frac{max \frac{w_k}{RTT_k}}{w_k} * RTT_k$; and k belongs to the set of subflow R.

BBR has been recently proposed and promoted by a group of Google researchers. They urge the need for changes in the fundamental design of TCP congestion control in order to make TCP perform better in the networks characterized by wireless links, bufferbloat, etc. BBR, therefore, attracts a lot of attention. It is a sender-based congestion control that does not require any modification from networks.

Instead of using the loss and ACK signals as the traditional congestion control, BBR aims to model an end-to-end communication over a complex network as a single bottleneck. BBR then uses an estimation for the available bottleneck data rate b_r and the minimal round-trip time RTT_{min} to determine the amount of sending data. It actually tries to calculate a path's available bandwidth delay product (bdp) through the bottleneck. The estimation of b_r is updated following the measurement of data delivered from the receiver. A BBR sender controls its transmission rate s_r with the help of pacing function and an estimated data rate b_r. In BBR, the limitation of in-flight data is set to 2bdp [13]. BBR probes for more bandwidth by increasing its sending rate s_r to $1.25s_{r0}$ for an RTT and directly reducing it again to $0.75s_{r0}$, where s_{r0} is the current estimate of the available data rate. The reduction aims at draining a potential queue that was possibly created by the higher rate. BBR uses a special *ProbeRTT* phase that tries to drain the queue completely in order to measure RTT_{min}. Different to the others, BBR is neither delay-based nor loss-based. Moreover, it ignores packet loss as congestion signal. It also does not explicitly react to congestion.

Note that, although BBR shows a lot of potential it is not perfect yet. There are ongoing debates as well as developments related to BBR. However, there are still several remaining issues (e.g. fairness) pointing out by the related work, which have not been solved yet. However, we mainly focus on the loss handling issue. We leave the other issues for future works.

C. Multipath TCP with BBR (mpbbr)

The trend in renovating the design of TCP congestion control, especially the way of handling loss in BBR, motivates us to investigate the advantageous feature. We aim to leverage those features for MPTCP in wireless networks. We first integrate the BBR implementation into the MPTCP kernel in a new implementation named Multipath TCP with BBR (i.e., *mpbbr*). It is expected that each subflow in *mpbbr* will be sent following an model-based algorithm that estimates the network's bottleneck on the associated path. Initially, we check the feasibility of realizing the real implementation of *mpbbr*. Although the idea is rather straightforward the integration process between MPTCP and BBR is not. The main reason is that the MPTCP kernel has been developed at a slower speed than the main kernel that includes BBR. We need to "downgrade" the code of BBR and compile it within the MPTCP kernel[1]. Thanks to the design of MPTCP, we could reuse and attach the BBR code to the MPTCP kernel, in which the BBR congestion control acts as an uncoupled congestion control.

III. Evaluation

A. Environment

To compare the performance of *mpbbr* and *balia*, we construct an indoor testbed, which includes a client and a server

[1]The recent release of MPTCP kernel (version 0.93) has included BBR.

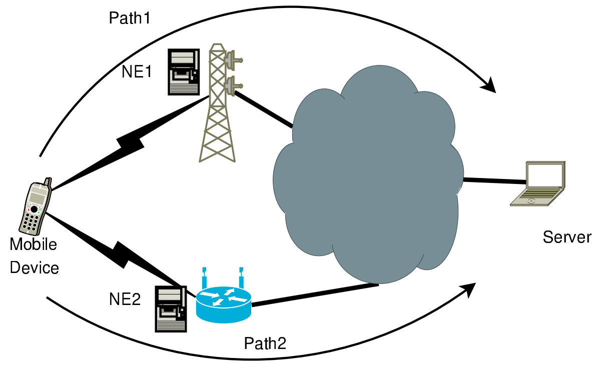


Fig. 2: Connection diagram in the testbed

TABLE I: Loss condition

	Loss (%)
Path1	(0.001, 0.01, 0.1, 1, 2, 5, 10)
Path2	(0.001, 0.01, 0.1, 1, 2, 5, 10)

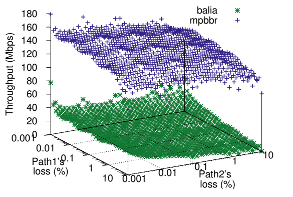
connecting via two paths. The testbed emulates a multipath wireless network (e.g., with LTE and Wi-Fi) that is similar to the one shown in Fig. 2. The client plays a role as a mobile device that communicates with an application server via two wireless links (i.e., Path1, Path2). On both the client and the server we use Linux kernel version 4.1.26 with the MPTCP version 0.91.0 [15], that includes the *balia* and our implementation of *mpbbr*. At each network configuration, an *iperf3* client generates a 60-second TCP flows to the server. We use the Linux network emulator *netem* (i.e., NE1, NE2 in the figure) to characterize the path parameters, which share the same values of bandwidth and delay (i.e., 100Mbps-bandwidth and 50ms-delay). In this evaluation, we want to compare the performance of two MPTCPs under different random loss conditions on two paths. We hence run experiments with all the combination of loss parameters shown in Table I.

B. Evaluation Results

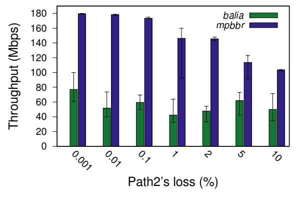
This section presents the evaluation results of *mpbbr* in a comparison to *balia*. The results with different combinations of two paths' loss condition are shown in Fig. 3, where Fig. 3a provides an overview of the average results. The detail values of throughput when the loss rate on Path1 is 0.001, 0.01, 0.1, 2, 5, 10 (%) are shown in Fig. 3b, Fig. 3c, Fig. 3d, Fig. 3e, Fig. 3f, Fig. 3g, Fig. 3h, respectively. In each figure, the Path2's loss value varies in the range shown in Table I. In the figures, the average, maximum, minimum values are collected and calculated at each loss condition.

Figure 3a provides a quick comparison of throughput performance of *mpbbr* and *balia*. We present the average values of throughput in the 3D curves. Since the distance between two curves are visible, we can easily observe that *mpbbr* outperforms *balia* in all investigated conditions. The details in the following figures also agree with the overview observation.

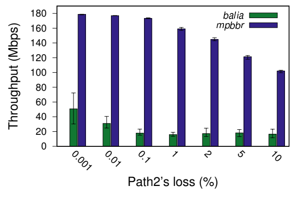
In general, *balia* performs not well when the two paths both experiences lost packets (i.e., the bad performance is expected due to the loss-based characteristic). In the case of 0.001% loss



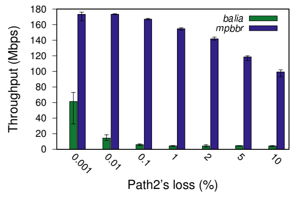
(a) Summary of all the results



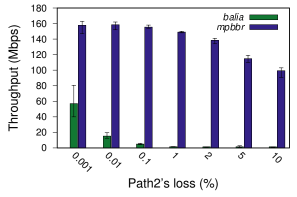
(b) 0.001% loss on Path1



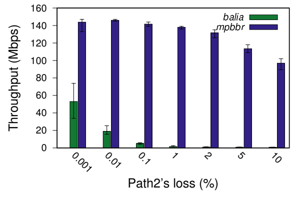
(c) 0.01% loss on Path1



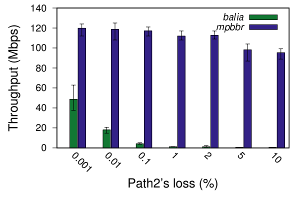
(d) 0.1% loss on Path1



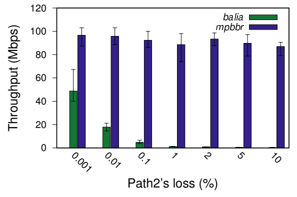
(e) 1% loss on Path1



(f) 2% loss on Path1



(g) 5% loss on Path1



(h) 10% loss on Path1

Fig. 3: Comparison of aggregated throughput of balia and mpbbr with different lossy conditions

in two paths, *balia* achieves the best throughput performance, however the value is still about half of the *mpbbr*'s. Another observation of *balia* throughput is that it is unstable in several cases. For example, Fig. 3b and Fig. 3c indicates that the throughput of *balia* with higher loss values may be bigger than the one of lower values (5% to 1%). On the other hand, the *mpbbr* throughput is more stable after different runs in all conditions.

When the loss rate of Path2 is higher than 0.1%, the results in all the figures indicate that the performance of *balia* is bad. The values are significantly lower than the capacity of each path. The throughput values may be useless in the real-life scenario. Meanwhile, *mpbbr* is still efficient since its throughput is relatively high and usable. Not only that, the model-based algorithm lets the throughput aggregation via different paths be still possible in *mpbbr* under various lossy conditions. Even in the extreme cases of 10% loss rate on both paths, *mpbbr* still can convey the application data efficiently.

IV. Related Works

As a transport layer technology, MPTCP has been proven to be efficient in a wide range of networks. Some typical examples include datacenters [16], software defined wireless networks [8], [17], WiGig systems [18], etc. MPTCP attracts mobile vendors (e.g., Korean Telecom), electronic devices producer (e.g., Apple) because of its practical feasibility in bandwidth aggregation and handover. Moreover, there are active efforts in developing MPTCP on popular platforms (Android, iOS but also Linux [15], FreeBSD [19], etc.). MPTCP is expected to be widely adopted in the next generation of mobile wireless networks. Therefore, the MPTCP's efficiency in mobile wireless needs to be carefully investigated and optimized. There are several works on performance evaluation of MPTCP in mobile wireless networks. In [5], [20], MPTCP has been efficiently deployed on a mobile to do switchover between a Wi-Fi and 3G link. In [21], the authors aim to characterize Wi-Fi/3G/4G networks with MPTCP performance in the wild. The same work has been done in [22]. However, the two works show different conclusions of MPTCP comparing to TCP. It also means that the dynamicity of wireless networks largely affects MPTCP, more experiments are generally required. In [23], the authors show applying experimental methods is efficient in discovering the misbehavior of MPTCP designs and implementations.

Loss handling in MPTCP is an important issue when deploying MPTCP on wireless networks. However, most of the previous works rely on the behavior of MPTCP congestion controls for such cases. As previously mentioned, the state-of-the-art congestion controls in MPTCP actually share the similar behavior with TCP for the lost signal. Moreover, the TCP variants before BBR have not performed well in lossy conditions. In our previous work [24], we propose a loss-awareness scheduler for *balia*, which switches back and forth between MPTCP and TCP following the loss information. The loss-awareness mechanism guarantees the MPTCP performance equals or is better than the TCP performance over a single path. The mechanism works well with under the situations where one path experiences loss. However, when multiple paths experience lost packets, the *balia*-based mechanism does not work well. Different to the previous work, this work leverages the advantage of the model-based TCP congestion control for MPTCP for handling random loss.

V. Conclusion and Future Works

MPTCP has been emerging as a promising solution for bandwidth aggregation and soft handover of wireless links in the current and next generation of mobile wireless networks. However, like its predecessor TCP, MPTCP with its recent breeds (i.e., *balia*) does not perform well under the lossy condition of wireless networks. In this paper, we show the feasibility of leveraging the advantages in state-of-the-art TCP congestion control for MPTCP's efficiency in the loss environment. Specifically, we exploit the model-based approach of BBR congestion control, which controls the sending speed based on the estimation of bottleneck bandwidth and round trip time. BBR interprets loss signals differently (i.e., neither loss-based or delay-based), hence it may be beneficial for MPTCP. We first have implemented an integration between BBR and MPTCP in *mpbbr*. We then evaluate *mpbbr* in various lossy condition comparing to the MPTCP with *balia*. The evaluation results show that *mpbbr* performs well across multiple loss conditions in which the MPTCP *balia* does not even work. In the future, we plan to improve *mpbbr* to a coupled congestion control, which tries to pool the share resources fairly between different *mpbbr*'s subflow. Moreover, we also plan to make *mpbbr* more friendly with the existing TCP and MPTCP congestion controls.

Acknowledgment

This research was conducted under a contract of R&D for Expansion of Radio Wave Resources, organized by the Ministry of Internal Affairs and Communications, Japan. Additionally, the first author is supported by the Leading Initiative for Excellent Young Researchers (LEADER) program from the Ministry of Education, Culture, Sports, Science and Technology, Japan.

References

[1] Alan Ford, Costin Raiciu, Mark Handley, and Olivier Bonaventure. TCP Extensions for Multipath Operation with Multiple Addresses. RFC 6824, 2013.

[2] Olivier Bonaventure, Christoph Paasch, and Gregory Detal. Use Cases and Operational Experience with Multipath TCP. RFC 8041, 2017.

[3] Mptcp kernel for android. http://multipath-tcp.org/pmwiki.php/Users/Android. (Accessed: 2019 January).

[4] Use multipath tcp to create backup connections for ios. https://support.apple.com/en-us/HT201373. (Accessed: 2019 January).

[5] Christoph Paasch, Gregory Detal, Fabien Duchene, Costin Raiciu, and Olivier Bonaventure. Exploring Mobile/WiFi Handover with Multipath TCP. In *Proc. ACM SIGCOMM CellNet*, pages 31–36, 2012.

[6] Y. s. Lim, Y. C. Chen, E. M. Nahum, D. Towsley, and K. W. Lee. Cross-layer Path Management in Multi-path Transport Protocol for Mobile Devices. In *Proc. IEEE INFOCOM*, pages 1815–1823, 2014.

[7] Kien Nguyen, Yusheng Ji, and Shigeki Yamada. A Cross-layer Approach for Improving WiFi Performance. In *Proc. IEEE IWCMC*, pages 458–463, 2014.

[8] Kien Nguyen, Kentaro Ishizu, and Fumihide Kojima. An evolvable, scalable, and resilient control channel for software defined wireless access networks. *Computers & Electrical Engineering*, 57(Supplement C):104 – 117, 2017.

[9] Costin R., Christoph P., Sebastien B., Alan F., Michio H., Fabien D., Olivier B., and Mark H. How Hard Can It Be? Designing and Implementing a Deployable Multipath TCP. In *Proc. USENIX NSDI*, 2012.

[10] R. Khalili, N. Gast, M. Popovic, and J.-Y. Le Boudec. MPTCP Is Not Pareto-Optimal: Performance Issues and a Possible Solution. *IEEE/ACM ToN*, 21(5):1651–1665, 2013.

[11] Yu Cao, Mingwei Xu, and Xiaoming Fu. Delay-based Congestion Control for Multipath TCP. In *Proc. 20th IEEE ICNP*, pages 1–10.

[12] Q. Peng, A. Walid, J. Hwang, and S.H. Low. Multipath tcp: Analysis, design, and implementation. *IEEE/ACM ToN*, PP(99), 2015.

[13] Neal Cardwell, Yuchung Cheng, C. Stephen Gunn, Soheil Hassas Yeganeh, and Van Jacobson. Bbr: Congestion-based congestion control. *ACM Queue*, 14, September-October:20 – 53, 2016.

[14] M. Becke, T. Dreibholz, H. Adhari, and E. P. Rathgeb. On the fairness of transport protocols in a multi-path environment. In *Proc. IEEE ICC*, pages 2666–2672, June 2012.

[15] S. Barre C. Paasch et al. Multipath TCP in the Linux Kernel, available from. http://www.multipath-tcp.org. (Accessed: 2019 January).

[16] Costin Raiciu, Sebastien Barre, Christopher Pluntke, Adam Greenhalgh, Damon Wischik, and Mark Handley. Improving Datacenter Performance and Robustness with Multipath TCP. In *Proc. ACM SIGCOMM*, pages 266–277, 2011.

[17] K. Nguyen, K. Ishizu, H. Murakami, F. Kojima, and H. Yano. A Scalable and Robust OpenFlow Channel for Software Defined Wireless Access Networks. In *Proc. IEEE VTC-Fall*, pages 1–5, 2015.

[18] K. Nguyen, M. G. Kibria, K. Ishizu, and F. Kojima. Feasibility Study of Providing Backward Compatibility with MPTCP to WiGig/IEEE 802.11ad. In *Proc. IEEE VTC Fall*, 2017.

[19] Multipath TCP for FreeBSD. http://caia.swin.edu.au/newtcp/mptcp/. (Accessed: 2019 January).

[20] Costin Raiciu, Dragos Niculescu, Marcelo Bagnulo, and Mark James Handley. Opportunistic Mobility with Multipath TCP. In *Proc. 6th ACM MobiArch*, pages 7–12, 2011.

[21] Yung-Chih Chen, Yeon-sup Lim, Richard J. Gibbens, Erich M. Nahum, Ramin Khalili, and Don Towsley. A Measurement-based Study of MultiPath TCP Performance over Wireless Networks. In *Proc. ACM IMC*, pages 455–468, 2013.

[22] Shuo Deng, Ravi Netravali, Anirudh Sivaraman, and Hari Balakrishnan. WiFi, LTE, or Both?: Measuring Multi-Homed Wireless Internet Performance. In *Proc. ACM IMC '14*, pages 181–194, 2014.

[23] Christoph Paasch, Ramin Khalili, and Olivier Bonaventure. On the Benefits of Applying Experimental Design to Improve Multipath TCP. In *Proc. ACM CoNEXT '13*, pages 393–398, 2013.

[24] Kien Nguyen, Gabriel Porto Villardi, Mirza Golam Kibria, Kentaro Ishizu, Fumihide Kojima, and Hiroyuki Shinbo. An enhancement of multipath TCP performance in lossy wireless networks. In *Proc. IEEE LCN Workshops*, pages 187–191, 2016.