Coupled Multipath BBR (C-MPBBR): A Efficient Congestion Control Algorithm for Multipath TCP

IMTIAZ MAHMUD, TABASSUM LUBNA, YEONG-JUN SONG, AND YOU-ZE CHO, (Senior Member, IEEE)
School of Electronics Engineering, Kyungpook National University, Daegu 41566, South Korea
Corresponding author: You-Ze Cho (yzcho@ee.knu.ac.kr)

This work was supported in part by the Next-Generation Information Computing Development Program through the National Research Foundation of Korea (NRF) funded by the Ministry of Science and ICT (MSIT) under Grant NRF-2017M3C4A7083676, in part by the Basic Science Research Program through the National Research Foundation of Korea (NRF) funded by the Ministry of Education under Grant NRF-2018R1A6A1A03025109, and in part by the National Research Foundation of Korea (NRF) grant funded by the Korea Government (MSIT) under Grant NRF-2019R1A2C1006249.

ABSTRACT

Multipath transmission control protocol (MPTCP) is a promising transport layer protocol that enables a device to utilize multiple communication interfaces simultaneously, thereby achieving high throughput. A congestion control algorithm (CCA) employed in MPTCP constitutes a key part that controls the data flow through different subflows (SFs). There are two fundamental challenges associated with MPTCP CCAs. First, MPTCP flows should have an advantage over single-path flows; second, MPTCP flows should be fair, indicating that SFs sharing a common bottleneck should occupy a capacity fairly close to that occupied by a single-path flow. Several MPTCP CCAs have been developed; however, they have failed to satisfy these challenges in all scenarios. Recently, Google has introduced the bottleneck bandwidth and round-trip-time (BBR), a new CCA for single-path TCP, achieving high throughput with minimum delay by employing a network model. In the present paper, we propose a novel MPTCP CCA based on BBR named coupled multipath BBR (C-MPBBR) that satisfies the fundamental challenges by exploiting the concept of network modeling in BBR. C-MPBBR addresses the first challenge by closing the low-bandwidth SFs by tracking the delivery rate and bottleneck bandwidth (BtlBW). Then, it satisfies the second challenge through identifying those SFs that share a common bottleneck and dividing the BtlBW share corresponding to a SF among them. We implemented C-MPBBR in the Linux kernel, tested it on a wide range of scenarios by the Mininet emulation experiments, and the real-world Internet, and confirmed that the proposed C-MPBBR outperforms the existing MPTCP CCAs in terms of successfully satisfying the fundamental challenges by ensuring both throughput and fairness.

INDEX TERMS

Multipath TCP, multipath BBR, coupled multipath BBR, coupled congestion control.

I. INTRODUCTION

The techniques facilitating the use of devices with multiple communication interfaces, such as 4G/5G and WiFi, have greatly advanced recently. It is anticipated that the simultaneous use of multiple interfaces will substantially improve the Internet experience, specifically, the quality of service (QoS) [1]. Although the existing transmission control protocol (TCP) does not support the concurrent use of multiple interfaces [2], an extension of TCP referred to as multipath TCP (MPTCP) [3] has attracted increasing attention as it enables multipath support by introducing small modifications in the transport layer. MPTCP defines a path between a pair of interfaces corresponding to two MPTCP terminals as a subflow (SF), i.e., each of the paths is designated as SF.

Congestion control (CC) is one of the key mechanisms in MPTCP as it enables avoiding network congestion by controlling the amount of transmitted traffic over each SF based on the network condition. There are two fundamental challenges associated with MPTCP CC [4]–[6]:

• Goal 1: To improve throughput, MPTCP should perform at least as a single-path flow performs on the best available path; in other words, MPTCP should always ensure an incentive over single-path flows.
• Goal 2: Provide fairness in terms of bandwidth (BW) usage. When sharing a common bottleneck, the total capacity occupied by the MPTCP SFs sharing that
bottleneck should be fairly close to that of a single-path flow; in other words, MPTCP flows should be fair.

Several MPTCP CC algorithms (CCAs), such as LIA [6], OLIA [7], BALIA [8], D-LIA [9], and Couple+ [10], have been proposed to address the fundamental challenges. However, they mainly focus on the second goal, which leads to the incapability of satisfying the first goal in most scenarios [11]. Moreover, their strategies mainly rely on modifying the legacy CCA in TCP (for example, Reno [12]) to fit MPTCP. As a result, the existing CCAs inherit the problems of the conventional loss-based CCAs. In addition, the issues get multiplied due to the multipath management of MPTCP. For example, a packet loss in a SF may often result in the severe degradation of performance in other SFs [13], [14].

Recently, Google has proposed the bottleneck bandwidth and round-trip-time (BBR) algorithm for single-path TCP, a new CCA that is aimed at avoiding congestion and packet losses by actively preventing persistent queue formation at the bottleneck [15]. BBR is used to ensure full BW utilization with lower delay. BBR not only controls the congestion window (CWND) but also regulates the sending rate to establish a more stable data transmission environment even at the presence of packet losses. BBR persuades a network model by sequentially measuring the bottleneck BW (BtlBW), minimum round-trip time (minRTT), and the delivery rate (DelRt). It identifies the sending rate based on the network model. Google reported [15] that BBR allowed reducing the YouTube’s median RTT by 80%, improved BW of B4 [16] by 133 times, and reached the Kleinrock’s optimal operating point [17]. Therefore, it is anticipated that the proper implementation of BBR in MPTCP can considerably enhance the performance of MPTCP.

Inspired by the promising performance of BBR in single-path TCP, several research groups developed different multipath BBR implementations [18]–[21]. However, in [18], [19], an uncoupled approach focused mainly on Goal 1 (Improve throughputs) was introduced, whereas in [20], [21], a coupled method concentrating on Goal 2 (Fairness) was implemented. Moreover, the coupled algorithms introduced in [20]–[22] were mainly grounded on the MPTCP’s existing coupled CCAs. As a result, they partially inherited the disadvantages of the MPTCP CCAs. Therefore, none of them satisfied the two fundamental goals of MPTCP. Although BBR could be used to represent a brief overview on a network by estimating BtlBW and minRTT, none of the proposed approaches implied utilizing these advantages.

Concerning Goal 2, those MPTCP flows that share the same bottleneck should occupy a total capacity fairly close to that of a single-path TCP flow. A simple way to realize it is to identify the SFs that share an identical bottleneck and divide the available capacity among them. To accomplish this, a key factor is to successfully identify a common bottleneck. BBR has a unique mechanism to sequentially measure BtlBW that constitutes the BW of a bottleneck router of that flow. In the case of proper application, this can be helpful in identifying a common bottleneck.

Concerning Goal 1, MPTCP always needs to provide the benefit over single-path TCP, or at least, it should not perform worse. BBR provides a key advantage here as well. BBR enables a mechanism to continuously measure DelRt that is a measure of the utilized capacity of an underlying network. An efficient implementation of this measurement along with the measured BtlBW can be useful to address Goal 1.

In the present work, we propose the coupled multipath BBR (C-MPBBR), a novel MPTCP CCA that is aimed to satisfy the two fundamental challenges of MPTCP by exploiting the available properties of BBR. The main contributions of this study can be summarized below:

1. Based on the sequentially measured BtlBW in BBR, C-MPBBR introduces a novel method to identify those SFs that share a common bottleneck.
2. C-MPBBR implies measuring BW that is available to a single SF going through a common bottleneck and then, distributing that BW among all SFs sharing that common bottleneck, thereby ensuring the fairness with regard to single-path flows.
3. Performing the continuous observation on DelRt and BtlBW, C-MPBBR regularly measures whether there is a benefit of using a multipath connection. If it finds that the multipath flows lag with regard to single-path ones, it closes a SF with the lowest BW. If a lag still appears, it closes the next SF with the lowest BW and so on until only one SF with the best BW remains. In this way, C-MPBBR focuses only on the best BW path and behaves similarly as a single-path flow. Therefore, C-MPBBR can satisfy Goal 1.
4. Finally, we implement MPBBR in the Linux kernel, and by conducting extensive real-world and Mininet emulation experiments in a wide range of scenarios, we have demonstrated that C-MPBBR successfully satisfies the two fundamental challenges associated with MPTCP CCAs, being a BBR-based CCA. Moreover, we compare its performance with other MPTCP CCAs, such as LIA, OLIA, BALIA, as well as the coupled multipath BBR proposed by Han et al. [21], and the uncoupled multipath BBR proposed by Nguyen et al. [18]. C-MPBBR not only outperforms its competitors in terms of throughputs in most cases but also ensures better fairness with respect to single-path flows.

The rest of the paper is organized as follows. Section II describes the related works. Section III discusses the basic principles of BBR in short and describes the C-MPBBR algorithm in detail. Section IV presents the comparison of the performance of C-MPBBR with those of considered CCAs. Finally, Section V concludes the paper.

II. RELATED WORKS

In this section, we briefly describe several previously proposed multipath BBR implementations and their limitations.

Nguyen et al. considered realizing an uncoupled multipath BBR in a Linux kernel [18]. They simply used the single-path BBR in multipath scenarios without any changes...
in the single-path algorithm and evaluated its performance in various scenarios with different packet losses. They concluded that compared with BALIA, their proposed multipath BBR demonstrated acceptable performance in all scenarios. However, they did not focus on the second fundamental challenge of MPTCP, namely, on the fairness with regard to single-path flows.

Zhang et al. introduced the other version of the uncoupled multipath BBR [19]. They claimed that the original BBR implementation relied on aggressive pacing to measure BtlBW, thereby causing delays in real-time video streaming. They considered using the different values to define the pacing rate in BBR, as well as a mechanism to decrease the pacing rate further when the bottleneck buffer becomes full. Moreover, they implemented a packet scheduling mechanism. However, as they mainly focused on performance improvement in the case of real-time video streaming, they did not consider the two fundamental challenges of MPTCP.

Zhu et al. proposed a coupled multipath BBR named wBBR [20]. They observed that their method achieved better performance compared with the other MPTCP CCAs; however, they mainly focused on satisfying the “congestion equality principle” [23] that did not correspond to the basic fundamental challenges of the MPTCP coupled CCAs, as proposed by IETF [6].

Han et al. proposed a coupled multipath BBR implementation focused on the fairness improvement [21]. They introduced a multiplication factor related to the BW of each SF. They conducted the experiments on the coupled BBR in a simple lossy environment and achieved better results compared with LIA, OLIA, and BALIA. However, it was unclear how coupled BBR would perform in the case of more complex scenarios and how it could always ensure at least equal or better throughput compared single-path flows; in other words, how it could realize Goal 1.

III. COUPLED MULTIPATH BBR (C-MPBBR)

In this section, we provide a brief description of the key mechanism of BBR, discuss the simple analysis of its behavior through a simple experiment, and then, we briefly describe proposed C-MPBBR CCA. We aim to address the two fundamental challenges of MPTCP. We start with describing the approach proposed to accomplish the Goal 2, then we explain the method to fulfill the Goal 1.

A. BRIEF OVERVIEW OF BBR

BBR is aimed at fully utilizing an underlying network while avoiding persistent queue formation at a bottleneck. Accordingly, it allows providing high throughput with minimum delay. A bottleneck is defined as a cluster in a network that has the lowest BW in a path between a sender and a destination. During each data connection, there is a bottleneck that determines the available BW, as well as the throughput of a network. Moreover, a persistent queue builds up here. BBR implies measuring BtlBW and minRTT periodically to actively control the flow of data. BBR mainly works in four following states: Startup — BBR starts with an exponential search for BtlBW until DelRt stops increasing; Drain — drains the excess data from the network that it poured during the Startup state; Probe Bandwidth (ProbeBW) — it consists of eight cycles, a specific ratio of the extra data (determined according to the pacing rate) is inputted into the network to check whether there is a new measure of BtlBW, and then, these extra data are drained during the following cycle, while during the next six cycles, BBR sends the data at a rate based on newly measured BtlBW. Probe RTT (ProbeRTT) — BBR updates the minRTT value by releasing the bottleneck buffer by limiting the size of CWND to only four packets. BBR enters this state only if it does not find an update on minRTT during the past ten-second interval.

There are mainly three control parameters in BBR that are used to control the data flow: pacing rate, CWND, and send quantum. Based on the estimated value of BtlBW and minRTT, BBR sets the pacing rate, CWND, and send quantum to efficiently use an underlying network.

Recently, the extensive research has been conducted to further improve the performance of BBR [24]–[30]. Moreover, Google has been actively working on developing BBR v2, an improved version of the current BBR v1 [31]. In the present work, we aim to introduce C-MPBBR on the basis of the current BBR v1. The recent improvements of BBR v1 can be easily incorporated into C-MPBBR. In addition, C-MPBBR can serve as a basis for the implementation of the upcoming BBR v2 in multipath scenarios in the near future.

B. SIMPLE EXPERIMENTAL ANALYSIS OF BBR

Fig. 1(a) illustrates a simple experimental setup deployed to test BBR. In this experiment, there were two senders and two destinations. Each sender initiated a data flow to a destination; therefore, there were two flows of data. CCA was defined as BBR. Both flows were started at the same time. Data transmission was executed for 120 seconds. Both flows shared a common bottleneck with the BW of 10 Mbps, the queue length of 2 BDP, and the random packet loss rate of 0.1%. RTT was the same in both flows. The Mininet [32] emulator was used for the experiment.

Fig. 1(b) represents the CWND of BBR in this simple scenario. BBR started with the exponential growth and drained the excess queue that was created to measure BtlBW, and completed the Startup and Drain states. Next, it continued the execution launching the ProbeBW state. It also entered into the ProbeRTT state periodically.

Fig. 1(c) shows the estimated BtlBW in both BBR flows that was accumulated and updated during the ProbeBW states. Although there were particular small glitches, both flows measured BtlBW almost equally. This was the reason underlying the almost equal CWND in both flows. Notably, the estimated BtlBWs in both flows were quite high than the real BtlBWs. Specifically, according to Fig. 1(a), the bottleneck link had the BW of 10 Mbps; when divided into the two flows, it allocated 5 Mbps to each flow. However, the average BW
FIGURE 1. A simple experimental analysis of BBR in terms of CWND, estimated BW, and throughput.

![Diagram](image)

A simple scenario in which two BBR flows share a common bottleneck.

Notable results were observed when we analyzed the throughput curve, as shown in Fig. 1(d). The average throughput estimates corresponding to Flow #1 and Flow #2 were 3.6 Mbps and 3.62 Mbps, respectively, which was less by 40-50% compared with the measured BtlBW. The reason behind this could lie in the induced packet losses and bottleneck queue.

In summary, the key findings can be listed as follows:

- All BBR flows equally estimated BtlBW.
- The estimated BtlBW exhibited the direct relation with throughput. As DelRt is highly related to throughput, a relation between DelRt and BtlBW can be established.

C. C-MPBBR: FULFILLING GOAL 2

As stated by BBR [15] and confirmed during the simple experiment reported in the previous section, the BBR flows estimate the available BtlBW equally. We incorporate this feature of BBR to resolve the unfairness issues in MPTCP, i.e., to fulfill Goal 2.

Goal 2 implies that if two or more SFs travel through the same bottleneck, combinedly they should take a capacity fairly close to that of a single-path TCP flow. The capacity is closely related to the available BW. Goal 2 can be simply interpreted as follows: the BW occupied by all SFs sharing a common bottleneck should be close to that of a single-path TCP flow. BBR provides a unique feature to estimate the available BW, and the multiple BBR flows sharing a common bottleneck can equally detect BtlBW. If we can successfully identify SFs sharing a common bottleneck, we will be able to divide the available BW of a bottleneck among all the SFs sharing that bottleneck. Therefore, the SFs sharing the common bottleneck will take nearly the same capacity as that of a single-path BBR flow.

To achieve this, the first challenge is to identify the SFs sharing a common bottleneck. We assume that the key to resolve this issue lies in the equal measurement of BtlBW. As we have observed previously, the flows traveling through a common bottleneck have almost equal measures of BtlBW. Therefore, if two or more SFs have close measures of BtlBW, we can consider that these SFs share a common bottleneck. Moreover, considering the small glitches observed in Fig. 1(c), we propose incorporating an error of ±α%. In addition, to ignore false-positive detections, we suggest observing the SFs with the same BtlBW for at least three consecutive ProbeBW states before making the final decision.

In summary, if two or more SFs have similar BtlBW with the deviation equal to ±α% during three successive complete ProbeBW states, then those SFs are considered to share a common bottleneck. Thereafter, we define set A containing the SFs going through the common bottleneck. For SF \( i \in A \), we divide the measured BtlBW by the number of SFs passing through the common bottleneck as follows:

\[
\text{BtlBW}_i = \frac{\text{BtlBW}_i}{|A|} \tag{1}
\]

It should be noted that the first and second cycles of ProbeBW state will proceed normally with the measured full BtlBW. This enables C-MPBBR to continuously measure the correct BtlBW during whole transmission for all SFs. Moreover, we divide BtlBW and not CWND, as following BtlBW, BBR automatically decides CWND, send quantum, and the pacing rate.

Moreover, if BtlBW \( k \) of SF \( k \in A \) does not show a similar BtlBW compared with other SFs (SF \( i \in A \)) during three successive complete ProbeBW states, then this SF \( k \) is removed from set \( A \) and continues to send the data with its estimated BtlBW \( k \). This way, by continuously observing BtlBW, C-MPBBR can keep on updating set \( A \), i.e., the set of SFs sharing the common bottleneck. The entire method is summarized in Algorithm 1, where we first describe the algorithm in words and then present the pseudo-code accord-
Algorithm 1 C-MPBBR: Fulfilling Goal 2

Description in simple words:
Start C-MPBBR flows with BBR’s default setting if two or more SFs has ±α% similar BW for three successive complete ProbeBW states then those SFs are ∈ A
BtlBW_SFᵢ = BtlBW_SFᵢ/|A|
end if
Continue to observe ProbeBW state if SFᵢ ∈ A does not has ±α% similar BW with other SFs ∈ A for three successive ProbeBW states then SFᵢ ≠ A
end if

Pseudo Code:

\[ α = 20 \]
\[ \text{number}\_of\_SFs\_in\_btlneck = 0 \]
\[ \text{if } cmpbbr \rightarrow \text{mode} = \text{CMPBBR\_PROBE\_BW} \text{ and } \text{cmpbbr} \rightarrow \text{cycle}\_\text{index} = 3 \text{ then} \]
\[ \text{bw}\_\text{lower}\_\text{limit} = \text{bw}\_\text{of}\_SF_{\text{this}} - ((\text{bw}\_\text{of}\_SF_{\text{this}} \times α)/100) \]
\[ \text{bw}\_\text{upper}\_\text{limit} = \text{bw}\_\text{of}\_SF_{\text{this}} + ((\text{bw}\_\text{of}\_SF_{\text{this}} \times α)/100) \]
\[ \text{for all subflow } i \text{ do} \]
\[ \text{if } \text{bw}\_\text{of}\_SF_{i} ≥ \text{bw}\_\text{lower}\_\text{limit} \text{ and } \text{bw}\_\text{of}\_SF_{i} ≤ \text{bw}\_\text{upper}\_\text{limit} \text{ then} \]
\[ \text{number}\_of\_SFs\_in\_btlneck++ \]
\[ \text{end if} \]
\[ \text{end for} \]
\[ \text{if } \text{number}\_of\_SFs\_in\_btlneck > 1 \text{ and } \text{cmpbbr} \rightarrow \text{last}\_\text{number}\_of\_SFs\_in\_btlneck > 1 \text{ then} \]
\[ \text{final}\_\text{number}\_of\_SFs\_in\_btlneck = \text{number}\_of\_SFs\_in\_btlneck} \]
\[ \text{else if } \text{number}\_of\_SFs\_in\_btlneck = 1 \text{ and } \text{cmpbbr} \rightarrow \text{last}\_\text{number}\_of\_SFs\_in\_btlneck > 1 \text{ then} \]
\[ \text{final}\_\text{number}\_of\_SFs\_in\_btlneck = \text{cmpbbr} \rightarrow \text{last}\_\text{number}\_of\_SFs\_in\_btlneck} \]
\[ \text{else} \]
\[ \text{final}\_\text{number}\_of\_SFs\_in\_btlneck = 1 \]
\[ \text{end if} \]
\[ \text{cmpbbr} \rightarrow \text{last}\_\text{number}\_of\_SFs\_in\_btlneck = \text{number}\_of\_SFs\_in\_btlneck} \]
\[ \text{end if} \]
\[ \text{bw}\_\text{of}\_SF_{\text{this}} = \text{bw}\_\text{of}\_SF_{\text{this}}/\text{final}\_\text{number}\_of\_SFs\_in\_btlneck} \]

D. C-MPBBR: FULFILLING GOAL 1

Goal 1 states that MPTCP flows should provide an incentive over single-path flows or at least ensure the comparably good performance. Therefore, if MPTCP flows cannot provide an incentive or perform at least equally to single-path flows, then it is preferable to stop multipath transmission and use a single-path only. As discussed in Section III.B, the BBR flows exhibit the relation between BtlBW and DelRt. We consider that this can be a key to resolve Goal 1.

It is a well-known fact that the BW of a network represents its available capacity. By estimating BtlBW, BBR computes the network capacity. Therefore, it is also expected that DelRt should be close to the measured BtlBW. We propose to achieve Goal 2 by ensuring the appropriate utilization of these two parameters.

To fulfill Goal 2, we propose to convert multipath flows into a single-path flow in the cases when the multipath flows provide no benefit over a single-path one. However, rather than stopping all SFs at once, we propose to follow a step-by-step procedure. Let Total_Del_Rt be the total DelRt gained by all SFs and highest_bw_among_all_SFṣ be the highest BW among all available SFṣ. C-MPBBR continuously observes BtlBW and DelRt. Considering five consecutive complete ProbeBW states (full cycle), if the following condition is true:

\[ \text{Total}_\text{Del}\_\text{Rt} < β \times \text{highest}_\text{bw}\_\text{among}\_\text{all}_\text{SFṣ} \]

then C-MPBBR stops the SF with the lowest BW and continues to monitor BtlBW and DelRt. If it finds that DelRt does not improve and Eq. (2) is satisfied for five consecutive ProbeBW states, it closes the next lowest BW SF. This is repeated until only the highest BW SF is left. Thereafter, C-MPBBR transforms into a single BBR flow and works following single-path BBR. Here, β is set equal to 40 following the gap between the estimated BW and DelRt, as observed in Fig.I. Moreover, C-MPBBR waits for five consecutive complete ProbeBW states so that it can eliminate the chances of false-positive results before closing SF. Algorithm 2 summarizes the procedure to achieve Goal 1.

IV. PERFORMANCE EVALUATION OF C-MPBBR

In this section, we discuss the evaluation of the performance of C-MPBBR concerning a wide range of scenarios which are designed to evaluate the specific properties of MPTCP CCAs. Moreover, we compare the performance of C-MPBBR with the conventional MPTCP CCAs, such as LIA [6], OLIA [7], and BALIA [8], as well as the recent multipath BBR implementations, including the ones reported by Han et al. (referred to as Han’s MPBBR) [21] and Nguyen et al. (referred to as U-MPBBR) [18]. We start with a detail description of the experimental setup and scenarios, and then compare the performance of the considered CCAs in the studied scenarios. Subsequently, we evaluate the performance of the considered CCAs in a complex network scenario to observe how they perform when different challenges are required to be addressed simultaneously.

In terms of the implementation in Linux kernel. Here, we set the value of α to 20. Moreover, the interested readers are encouraged to access and test the implemented Linux Kernel code for C-MPBBR uploaded in the GitHub repository given in [33].
We executed at least 30 experiments for each test case. The experiment time was 120 seconds. As we aimed at presenting an actual CWND and throughput obtained in the experiments, we demonstrated the results of randomly selected emulation experiments in Figs. 4-10. Figs. 11-15 demonstrate the mean, standard deviation, median, 25% and 75% percentiles, and the degree of dispersion.

A. EXPERIMENTAL SETUP

We conducted the performance evaluation through emulation experiments based on the Linux network namespaces defined in the Mininet [32] emulator. Here, “fq” [34] was enabled as the queueing discipline; “NetEm” [35] and “ethtool” [36] were used to configure RTT and BW, respectively; “iperf3” [37] was employed to transmit the data between a client and a server and to assess the total throughput; “ifstat” [38] was used to measure the throughput per flows; “tcprove” [39] was utilized to measure CWND and other internal parameters of BBR. MPTCP v0.93.4 deployed in Linux kernel v4.9.169 was used to conduct the experiments.

B. DIFFERENT SCENARIOS TO OBSERVE SPECIFIC PROPERTIES OF MPTCP CCAs

Fig. 2 represents the experimental scenarios considered to evaluate and compare the performance of C-MPBBR. In Scenario #1, as shown in Fig. 2(a), the client and server were connected via two different links having different properties. No background traffic was introduced in this scenario. This scenario was defined to understand what part of the whole capacity C-MPBBR and considered MPTCP CCAs could utilize in an underlying network. Scenario #2 (Fig. 2(b)) was designed to observe the performance of multipath flows when one SF shared the bottleneck with a single-path flow. In Scenario #3 (Fig. 2(c)), the aim was to observe the performance of the MPTCP CCAs in terms of addressing Goal 2; namely, how they performed when two SFs shared a common bottleneck while competing with a single-path flow sharing the same bottleneck. Scenario #4 (Fig. 2(d)) represented an interesting case in which SF-1 passed through a high BW path with small delay and low losses, whereas SF-2 passed through comparatively a rather narrow BW path with long delay and high losses. Moreover, SF-2 shared its path with a single-path flow. This allowed observing how differently MPTCP CCAs addressed this tricky condition in which a single-path flow received an upper-hand over the multipath flow. Finally, Scenario #5 (Fig. 2(e)) was specifically designed to observe the shortcomings of C-MPBBR. Here, the client and server were connected through two identical paths. C-MPBBR considers the SFs with the equal BtlBW as those sharing the same bottleneck. In Scenario #5, this mechanism could force C-MPBBR to consider the two SFs going through different paths as SFs sharing a common bottleneck.

Finally, we complete the evaluation by observing their performance in the real-world Internet.

Algorithm 2 C-MPBBR: Fulfilling Goal 1

**Description in simple words:**

Start C-MPBBR flows

while $SF_{having \; lowest \; bw} \neq SF_{having \; highest \; bw}$

- if Total_Del_RT $< \beta \times \text{highest\;bw\;among\;all\;SFs}$ for five successive complete ProbeBW states then
  - Stop: $SF_{having \; lowest \; bw}$
  - Break

else

- Break

end if

end while

**Pseudo Code:**

- $\beta = 40$
- Total_Del_RT = 0
- lowest_bw_among_all_SF = 999999999
- highest_bw_among_all_SF = 0
- total_number_of_SF = 0

if $\text{cmpbr} \rightarrow \text{mode} = \text{CMPBBR\_PROBE\_BW}$ and

for all subflow $i$ do

- Calculate: $\text{Del}_i\_\text{of}_i\_\text{SF}$
  - total_number_of_SF $++$
  - Total_Del_RT = Total_Del_RT + $\text{Del}_i\_\text{of}_i\_\text{SF}$
  - if lowest_bw_among_all_SF $>$ bw_of_SF then
    - lowest_bw_among_all_SF = bw_of_SF

end if

- if highest_bw_among_all_SF $<$ bw_of_SF then
  - highest_bw_among_all_SF = bw_of_SF

end if

end for

end if

threshold_bw_for_stopping_lowest_bw_SF = highest_bw_among_all_SF = ($(\text{highest\;bw\;among\;all\;SFs} \times \beta) / 100$)

if threshold_bw_for_stopping_lowest_bw_SF $>$ Total_Del_RT and $\text{cmpbr} \rightarrow \text{last\;number\;of\;SFs\;in\;btlneck} < 2$ and total_number_of_SF $> 1$

and lowest_bw_among_all_SF $\neq \text{highest\;bw\;among\;all\;SFs}$ then

- $\text{cmpbr} \rightarrow \text{stop\;lowest\;bw\;SF\;count}++$

else

- $\text{cmpbr} \rightarrow \text{stop\;lowest\;bw\;SF\;count} = 0$

end if

if $\text{cmpbr} \rightarrow \text{stop\;lowest\;bw\;SF\;count} \geq 5$ and total_number_of_SF $> 1$ and bw_of_SF_this $= lowest_bw_among_all_SF$ then

- $\text{cmpbr} \rightarrow \text{stop\;lowest\;bw\;SF\;count} = 5$

Close: SF_this

end if
C. PERFORMANCE EVALUATION IN TERMS OF CONGESTION WINDOW AND THROUGHPUT

Fig. 3 represents the CWND and throughput per flow for the considered MPTCP CCAs observed in Scenario #1. In this scenario, all BBR-based MPTCP CCAs outperformed all loss-based MPTCP CCAs. The loss-based MPTCP CCAs, such as LIA, OLIA, and BALIA, were greatly affected by the inability to properly utilize an underlying network for both SFs. During the 120 seconds emulation time, the average throughput for C-MPBBR was equal to 11.7 Mbps, whereas that of LIA was only 8.8 Mbps. The induced packet losses in Scenario #1 caused LIA to slow down the sending rate. Moreover, the aggressive fairness algorithm of LIA caused the throughput being suppressed further and resulted in poor performance [11]. The same cause applied to OLIA and BALIA as well. However, as BtBW was different, C-MPBBR could understand that no SFs shared a common bottleneck, thereby enabling both SFs to behave as two separate BBR flows and to utilize the BW in full. U-MPBBR algorithm also performed similarly. Although, Han’s MPBBR lacks such mechanism to identify bottleneck sharing SFs, they performed considerably well because of the congestion avoidance nature of BBR.

Fig. 4 shows the CWND and throughput of the considered CCAs for Scenario #2. Please note that the background traffic for SF-1 was BBR for C-MPBBR, Han’s MPBBR, and U-MPBBR, as well as Reno for LIA, OLIA, and BALIA. For clarity, we named them as C-MPBBR vs. BBR, LIA vs. Reno, and so on. In this scenario, the paths of SF-1 and SF-2 had the BW of 10 Mbps and 5 Mbps, respectively. Because there was background traffic in the path of SF-1, by considering fair BW sharing principle, SF-1 would receive the BW approximately 5 Mbps. In this scenario, when C-MPBBR SFs passed through two different paths, they behaved like two separate BBR flows. Therefore, following the principle of a BBR flow, SF-1 shared the BW with the single-path BBR flow and had the average throughput of 3.7 Mbps, whereas the single-path BBR flow received an average of 4.1 Mbps. Moreover, the total average throughput of C-MPBBR was 7.5 Mbps. Han’s MPBBR and U-MPBBR also exhibited a similar trend. However, in LIA, SF-1 achieved the average throughput of only 1.4 Mbps, and the total average throughput was 4.9 Mbps. Notably, the single-path Reno flow could achieve the average throughput of 7.4 Mbps, which was much higher than that of LIA. A similar trend in results was also observed for OLIA and BALIA. This indicated that C-MPBBR and other BBR-based MPTCP variants could efficiently utilize and share the underlying network, in contrast with LIA, OLIA, and BALIA. According to [11], we consider the cause to be an aggressive fairness ensuring mechanism used in LIA, OLIA, and BALIA.

Fig. 5 represents the performance of the considered CCAs when SF-1 competes with CUBIC in Scenario #2. A similar trend in the results was observed similarly as in the case of SF-1 competing with BBR/Reno.
Fig. 6 and Fig. 7 represent the performance of the confined MPTCP CCAs in Scenario #3 in which the background traffic corresponds to BBR/Reno and CUBIC, respectively. Here, both SFs traveled through a common bottleneck, and the bottleneck was shared with a single-path TCP flow. According to Goal 2, the two SFs combinedly occupy the capacity close to that of a single-path TCP flow. Here, among all BBR-based CCAs, only C-MPBBR successfully satisfied this criterion. The average throughput of C-MPBBR was 4.3 Mbps, whereas that of the single-path BBR flow was 3.8 Mbps, being fairly close to each other. Moreover, Goal 1 stated that multipath flows had to always ensure an incentive over a single-path flow, which was also addressed. On the other hand, Han’s MPBBR’s and U-MPBBR’s average throughput was 6.1 and 5.7 Mbps, respectively. And their rival single-path BBR flow’s average throughput was
2.5 and 2.6 Mbps, respectively. The same trend in the results was observed while competing with CUBIC. Therefore, Goal 2 was violated as those algorithms were excessively greedy in terms of absorbing BW. On the contrary, LIA achieved the average throughput of 2.8 Mbps, and the single-path Reno reached that of 5.4 Mbps, meaning that LIA failed to hold its equal share. The same trend in the results was observed for LIA against CUBIC, and OLIA and BALIA against both Reno and CUBIC.

Fig. 8 and Fig. 9 illustrate the performance of the considered MPTCP CCAs while competing with BBR/Reno, and CUBIC in Scenario #4, respectively. Here, SF-1 went through a path with the BW of 50 Mbps, 1 ms delay, and 0.01% loss. Whereas, SF-2 traveled through a path with 1 Mbps BW, 200 ms delay, and 0.2% loss. Moreover, SF-2 shared its path with a single-path flow. In this scenario, a single-path flow through the path of SF-1 would get an upper-hand over the multipath flow because the properties of the path
I. Mahmud et al.: C-MPBBR: A Efficient CCA for Multipath TCP

FIGURE 9. In Scenario #4, CWND and throughput of: (a)-(b) C-MPBBR vs. CUBIC, (c)-(d) Han’s MPBBR vs. CUBIC, (e)-(f) U-MPBBR vs. CUBIC, (g)-(h) LIA vs. CUBIC, (i)-(j) OLIA vs. CUBIC, and (k)-(l) BALIA vs. CUBIC.

FIGURE 10. In Scenario #5, CWND and throughput of: (a)-(b) C-MPBBR vs. BBR, (c)-(d) Han’s MPBBR vs. BBR, (e)-(f) U-MPBBR vs. BBR, (g)-(h) LIA vs. Reno, (i)-(j) OLIA vs. Reno, and (k)-(l) BALIA vs. Reno.

of SF-2 would result in head-of-line blocking at the receiver side for the multipath flows. Both multipath and single-path flows would struggle to properly utilize the network capacity. Following the advanced algorithm, C-MPBBR successfully recognized it and terminated SF-2 after some time, as shown by the red dotted circle. Therefore, C-MPBBR could achieve the average throughput of 36 Mbps, and the single-path BBR flow achieved 0.86 Mbps. On the contrary, Han’s MPBBR and U-MPBBR reached only 15.4 and 16.5 Mbps, and their competitor single-path BBRs acquired 0.47 and 0.4 Mbps, respectively. The same trend in the results was observed for CUBIC. Moreover, in this scenario, LIA performed significantly better than Han’s MPBBR and U-MPBBR; it achieved the average throughput of 24.3 Mbps, and the single-path BBR flow acquired 0.8 Mbps. The same trend in the results was observed against CUBIC, for OLIA and BALIA. The success of LIA, OLIA, and BALIA in this scenario lied in the usage of an aggressive fairer algorithm [11].

D. PERFORMANCE EVALUATION IN TERMS OF AGGREGATE BENEFIT

Following [40], to thoroughly investigate the network utilization with regard to multipath flows, considering the goodput and available BW, we defined a parameter “Aggregate Benefit (Ag_bft)” as follows:

$$\text{Ag}_bft = \begin{cases} G_t - BW_{max}, & \text{if } G_t \geq BW_{max} \\ \sum_{m=1}^{n} BW_{m} - BW_{max}, & \text{if } G_t < BW_{max} \end{cases}$$

where $G_t$, $BW_{m}$, and $BW_{max}$ were the total goodput of multipath flows, the BW of path $m$, and the largest BW among
all paths, respectively. The better was the result of $A_g_{bft}$, the better was the performance.

Fig. 11 shows the performance of different CCAs in terms of $A_g_{bft}$. It should be noted that “vs. BBR/Reno” implies that the background traffic for C-MPBBR, Han’s MPBBR, and U-MPBBR corresponds to BBR; and LIA, OLIA, and BALIA to Reno; and “vs. CUBIC” means that the background traffic corresponds to CUBIC. This rule applies for all the scenarios unless stated otherwise. For Scenarios #1-2, in the case of the absence of background traffic, the performance estimates of the BBR-based MPTCP CCAs surpassed those of LIA, OLIA, and BALIA owing to the appropriate network modeling by BBR. Moreover, all BBR-based MPTCP CCAs resulted in almost equal $A_g_{bft}$. For Scenarios #2-4, the same trend in the results was observed for both cases, including “vs. BBR/Reno” and “vs. CUBIC”. Furthermore, the same trend in the results was observed while comparing the BBR-based CCAs to LIA, OLIA, and BALIA except for Scenario #4. In Scenario #4, C-MPBBR surpassed all other CCAs, as C-MPBBR adopted a unique mechanism to recognize the situations in which single-path flows performed better than multipath ones and to deal with such situations by converting itself to a single-path flow, thereby successfully addressing Goal 1. Moreover, LIA, OLIA, and BALIA also performed better than Han’s MPBBR and U-MPBBR in this scenario.

However, in Scenarios #2-3, $A_g_{bft}$ of C-MPBBR was slightly less than those of Han’s MPBBR and U-MPBBR. This was because the SFs were going through a shared path, to satisfy Goal 2, C-MPBBR reduced its BW so that single-path flows could obtain a fair share. This is discussed in detail in the subsequent section.

E. PERFORMANCE EVALUATION IN TERMS OF FAIRNESS TO SINGLE-PATH FLOWS

In the previous section, we observed how multipath flows utilized an underlying network. In this section, we discuss how fairly they behave with regard to single-path flows.

First, we analyzed what part of network capacity single-path flows could exploit. To clearly estimate performance, we calculated the normalized throughput ($Norm_{Thpt}$) as follows:

$$Norm_{Thpt} = \frac{Thpt}{Available_{BW}}$$

where $Thpt$ and $Available_{BW}$ denoted the throughput obtained by a single-path flow and the actual available BW for that flow, respectively. Notably, $Available_{BW}$ did not correspond to the available physical BW but to the fair BW available to each flow going through the bottleneck. The value of $Norm_{Thpt}$ equal to one, less than one, and greater than one represented that a single-path flow fully utilized the available BW, underutilized the available BW, and utilized above its fair share, respectively.

Fig. 12 represents $Norm_{Thpt}$ for single-path flows in Scenarios #2-4. Notably, the actual available BW for single-path flows in Scenarios #2-4 was 5, 3.3, and 0.5 Mbps, respectively. In Scenarios #2-3, compared with C-MPBBR, both the BBR and CUBIC flows reached the $Norm_{Thpt}$ value near one and performed significantly better with regard to Han’s MPBBR and U-MPBBR. However, the single-path Reno and CUBIC flows occupied BW more than their fair share while competing with LIA, OLIA, and BALIA. In Scenario #4, the single-path BBR and CUBIC flows achieved the highest $Norm_{Thpt}$ value competing with...
C-MPBBR. They received the BW approximately twice larger than their fair share. In Scenario #4, to ensure better throughput, C-MPBBR stopped SF-2 after some time. This allowed releasing the BW for single-path flows and enabled them to utilize the total BW of that path. This resulted in better throughput both for the multipath and single-path flows. Single-path flows showed a similar trend in the results competing with the other MPTCP CCAs, as observed in Scenarios #2-3.

Finally, for Scenarios #2-3, we calculated the Jain’s fairness index [41], [42], as shown in Fig. 13. Notably, we did not do that for Scenario #4, because, by design, C-MPBBR stopped SF-2 to ensure better throughput. This invalidated the fairness issue in this scenario.

Moreover, in Scenario #3, SF-1 and SF-2 shared the common bottleneck with a single-path TCP flow. Here, SF-1 and SF-2 together should take a BW fairly close to that of a single-path TCP flow, i.e., the application using MPTCP should occupy approximately equal or slightly more BW than the single-path TCP to fulfill both Goals. Therefore, during the fairness calculation of Scenario #3, we considered the total capacity used by the MPTCP application, and compared it to the capacity utilized by the single-path TCP application.

In addition, we conducted experiments to observe how fair C-MPBBR behaves when it competes with another C-MPBBR. We considered Scenarios #2-3 in the absence of background traffic. The multipath client started two “iperf3” [37] data flows with the multipath server setting C-MPBBR as the CCA. We measured the total capacity achieved by the two applications and calculated the Jain’s fairness index.

From Fig. 13, it is clear that the C-MPBBR ensured the best fairness index among all considered CCAs, while competing with BBR, CUBIC, and another C-MPBBR in both scenarios. As observed previously, Han’s MPBBR and U-MPBBR could acquire better $A_g,bft$ owing to their greedy BW absorbing nature. This ultimately resulted in blocking other flows in a shared bottleneck and violating Goal 2. However, due to the aggressive nature in terms of fairness, LIA, OLIA, and BALIA could not obtain a fair share for themselves, violating Goal 1 and resulting in a poor fairness index.
We noted that the best performance of C-MPBBR in terms of fairness could be attributed to its intelligent algorithm aimed to recognize a shared bottleneck, as well as to its fair BW allocation mechanism for such situations. This enabled C-MPBBR not only to ensure an appropriate BW share for itself, but also to the competing flows.

**F. PERFORMANCE EVALUATION IN A COMPLEX NETWORK SCENARIO**

In this section, we evaluated the performance of the considered CCAs in a complex network scenario, as shown in Fig. 14(a). In this scenario, the multipath client and server were connected via five different paths, denoted as SF-1 to SF-5, respectively. SF-2 and SF-3 as well as SF-4 and SF-5 traveled through two separate common bottlenecks. Moreover, background traffic was present in all the bottlenecks. The path of SF-1 had the highest BW and lowest delay. It was also shared with a single-path flow, thus the available fair BW for SF-1 became 10 Mbps. The key challenge for SF-1 would be achieving full utilization of the BW while fairly sharing the BW with the single-path flow. SF-4 and SF-5 shared a common bottleneck, having the BW of 20 Mbps. This bottleneck was also shared with a single-path flow leaving a fair share of 5 Mbps for each SF. Therefore, SF-4 and SF-5 would require to combine to take a BW equal to single-path flow which is around 10 Mbps. Finally, the common bottleneck shared between SF-2 and SF-3 had a very narrow BW of 1 Mbps, with a high delay of 200 ms, and packet loss of 0.3%. The fair share for each of the SFs was only 0.25 Mbps. Compared to the other SFs, packets traveling through SF-2 and SF-3 would often cause head-of-line blocking at the receiver. Therefore, it would be better to terminate these flows and leave this bottleneck for the single-path flow.

Fig. 14(b)-(c) show the $Ag_{bft}$ when the background traffic was BBR/Reno and CUBIC, respectively. Notably, C-MPBBR achieved the highest $Ag_{bft}$ due to its advanced algorithm that stops the SFs with the lowest BW to always ensure an incentive over single-path flows. Both Han’s MPBBR and U-MPBBR could not achieve such performance because they continued sending packets through SF-2 and SF-3, which caused the head-of-line blocking at the receiver. LIA, OLIA, and BALIA’s poor performance can be attributed to their aggressive bias toward fairness.

Fig. 14(d) represents the Jain’s fairness index for the complex network scenario. Here, we also measured the performance of C-MPBBR while it competed with another C-MPBBR flow in the absence of background traffic, following the same procedure as mentioned in Section IV(E). Again, C-MPBBR ensured the best fairness index among the considered CCAs because of the proposed intelligent algorithm. Han’s MPBBR and U-MPBBR showed poor fairness indices due to their greedy BW absorbing nature. Further, the poor performance of LIA, OLIA, and BALIA was because of their aggressive nature with regard to fairness.

**G. PERFORMANCE EVALUATION IN THE REAL-WORLD INTERNET**

Finally, we evaluated the performance of the considered CCAs in the real-world Internet. We set up a multipath client and server at the two ends of the Kyungpook National University, Daegu campus. The multipath server computer was equipped with an Intel Core i5-9600K 3.7GHz processor,
V. CONCLUSION

In the present paper, we considered the problem of developing an appropriate MPTCP CCA that can satisfy the fundamental challenges of MPTCP. Several existing MPTCP CCAs were deemed aggressive so that they failed to utilize an underlying network properly. Others were so aggressive that they blocked other flows in a shared bottleneck in terms of BW. To address this issue, we proposed C-MPBBR, a novel BBR-based CCA for multipath scenarios. C-MPBBR was designed to be easily implementable inside BBR v1. We deployed it in the Linux kernel and made the code available online.

We conducted the extensive emulation and real-world experiments concerning several critical scenarios and found that C-MPBBR could successfully satisfy the fundamental challenges of MPTCP. It managed to avoid being excessively greedy for BW or aggressive in terms of fairness by identifying an appropriate trade-off. C-MPBBR was capable of fully utilizing an underlying network by ensuring fairly high throughput for itself and other competing flows. Moreover, it provided an incentive over single-path flows while enabling a fair share for rivals.

In the future research work, C-MPBBR can be further improved and extended by exploiting the recent developments in BBR v1. Specifically, we consider that the proposed algorithm can serve as a basis for the implementation of the upcoming BBR v2 in multipath scenarios in the near future.

REFERENCES

[1] D. Ma and M. Ma, “A QoS oriented vertical handoff scheme for WiMAX/WLAN overlay networks,” IEEE Trans. Parallel Distrib. Syst., vol. 23, no. 4, pp. 598–606, Apr. 2012.
[2] M. Allman, V. Paxson, and E. Blanton, TCP Congestion Control, RFC, document 5681, Sep. 2009.
[3] A. Ford, C. Raiciu, M. Handley, and O. Bonaventure, TCP Extensions for Multipath Operation With Multiple Addresses, RFC, document 6824, 2013.
[4] W. Wei, K. Xue, J. Han, D. S. L. Wei, and P. Hong, “Shared bottleneck-based congestion control and packet scheduling for multipath TCP,” IEEE/ACM Trans. Netw., vol. 28, no. 2, pp. 653–666, Apr. 2020.
[5] C. Raiciu, D. Wischik, and M. Handley, “Practical congestion control for multipath transport protocols,” Univ. College London, London, U.K., Tech. Rep., 2009. [Online]. Available: https://pdfs.semanticscholar.org/46a4/a3291d27fe5f5466e40c6603059301095ab80.pdf
[6] C. Raiciu, M. Handley, and D. Wischik, Coupled Congestion Control for Multipath Transport Protocols, IETF RFC, document 6356, Oct. 2011.
[7] R. Khalili, N. Gast, M. Popovic, and J.-Y. Le Boudec, “MPTCP is not Pareto-optimal: Performance issues and a possible solution,” IEEE/ACM Trans. Netw., vol. 21, no. 5, pp. 1651–1665, Oct. 2013.
[8] Q. Peng, A. Walid, J. Hwang, and S. H. Low, “Multipath TCP: Analysis, design, and implementation,” IEEE/ACM Trans. Netw., vol. 24, no. 1, pp. 596–609, Feb. 2016.
[9] T. Lubna, I. Mahmud, and Y.-Z. Cho, “D-LIA: Dynamic congestion control algorithm for MPTCP,” ICT Express, 2020, doi: 10.1016/j.ictex.2020.03.005.
[10] X. Xue, J. Han, H. Zhang, K. Chen, and P. Hong, “Migrating unfairness among subflows in MPTCP with network coding for Wired–Wireless networks,” IEEE Trans. Veh. Technol., vol. 66, no. 1, pp. 798–809, Jan. 2017.
[11] R. Lubben and J. Morgenroth, “An odd couple: Loss-based congestion control and minimum RTT scheduling in MPTCP,” in Proc. IEEE 44th Conf. Local Comput. Netw. (LCN), Oct. 2019, pp. 300–307.
[12] T. Henderson, S. Floyd, S. Gurtov, and Y. Nishida, The NewReno Modification to TCP’s Fast Recovery Algorithm, document RFC 6582, Internet Engineering Task Force, Apr. 2012.
[13] S. Ferlin, S. Kucera, H. Claussen, and O. Alay, “MPTCP meets FEC: Supporting latency-sensitive applications over heterogeneous networks,” IEEE/ACM Trans. Netw., vol. 26, no. 5, pp. 2005–2018, Oct. 2018.
[14] S. R. Pokhrel, M. Panda, and H. L. Vu, “Analytical modeling of multipath TCP over last-mile wireless,” IEEE/ACM Trans. Netw., vol. 25, no. 3, pp. 1876–1891, Jan. 2017.
[15] N. Cardwell, Y. Cheng, C. S. Gunn, S. H. Yeganeh, and V. Jacobson, “BBR: Congestion-based congestion control,” Queue, vol. 14, no. 5, pp. 20–53, Oct. 2016.
[16] S. Jain, “B4: Experience with a globally-deployed software defined WAN,” ACM SIGCOMM Comput. Commun. Rev., vol. 43, no. 4, pp. 3–14, 2013.
[17] L. Kleinrock, “Power and deterministic rules of thumb for probabilistic problems in computer communications,” in Proc. Int. Conf. Commun., vol. 43, 1979, pp. 1–43.
[18] K. Nguyen, M. G. Kibria, K. Ishizu, F. Kojima, and H. Sekiya, “An evaluation of multipath TCP in lossy environment,” in Proc. IEEE Int. Conf. Persys. Comput. Commun. Workshops (PerCom Workshops), Mar. 2019, pp. 573–577.
C. A. Grazia, M. Klapez, and M. Casoni, “BBRp: Improving TCP BBR congestion control,” in Proc. IEEE 89th Veh. Technol. Conf. (VTC-Fall), Aug. 2018, pp. 1–5.

J. Han, K. Xue, Y. Xing, P. Hong, and D. S. L. Wei, “Measurement and redesign of BBR-based MPTCP,” in Proc. ACM SIGCOMM Conf. Posters Demos (SIGCOMM Posters Demos), 2019, pp. 75–77.

J. Han, Y. Xing, K. Xue, D. S. L. Wei, G. Xue, and P. Hong, “Leveraging coupled BBR and adaptive packet scheduling to boost MPTCP,” 2020, arXiv:2002.06284. [Online]. Available: http://arxiv.org/abs/2002.06284

Y. Cao, M. Xu, and X. Fu, “Delay-based congestion control for multipath TCP,” in Proc. 26th IEEE Int. Conf. Netw. Protocols (ICNP), Oct./Nov. 2012, pp. 1–10.

Y. Zhang, L. Cui, and F. P. Tso, “Modest BBR: Enabling better fairness for BBR congestion control,” in Proc. IEEE Symp. Comput. Commun. (ISCC), Jun. 2018, pp. 00646–00651.

M. Yang, P. Yang, C. Wen, Q. Liu, J. Luo, and L. Yu, “Adaptive-BBR: Fine-grained congestion control with improved fairness and low latency,” in Proc. IEEE Wireless Commun. Netw. Conf. (WCNC), Apr. 2019, pp. 1–6.

G.-H. Kim, I. Mahmud, and Y.-Z. Cho, “Fairness improvement of BBR congestion control algorithm for different RTT flows,” in Proc. Int. Conf. Electron., Inf., Commun. (ICEIC), Jan. 2019, pp. 1–2.

I. Mahmud, G.-H. Kim, T. Lubna, and Y.-Z. Cho, “BBR-ACD: BBR congestion control with advanced congestion detection,” Electronics, vol. 9, no. 1, p. 136, Jan. 2020.

I. Mahmud and Y.-Z. Cho, “BBR Advanced (BBR-A)—Reduced retransmissions with improved fairness,” ICT Express, 2020, doi: 10.1016/j.ictex.2020.05.009.

J. Wang, Y. Zheng, Y. Ni, C. Xu, F. Qian, W. Li, W. Jiang, Y. Cheng, Z. Cheng, Y. Li, X. Xie, Y. Sun, and Z. Wang, “An active-passive measurement study of TCP performance over LTE on high-speed rails,” in Proc. 25th Annu. Int. Conf. Mobile Comput. Netw., Aug. 2019, pp. 1–16, doi: 10.1145/3300061.3300123.

C. A. Grazia, M. Klapaz, and M. Casoni, “BBRp: Improving TCP BBR performance over WLAN,” IEEE Access, vol. 8, pp. 43344–43354, 2020.

N. Cardwell. (2019). BBR v2: A Model-based Congestion Control Update. Accessed: Jun. 10, 2020. [Online]. Available: https://datatracker.ietf.org/meeting/105/materials/slides-105-icrg-bbr-v2-a-model-based-congestion-control

B. Lantz, B. Heller, and N. McKeown, “A network in a laptop: Rapid prototyping for software-defined networks,” in Proc. 9th ACM SIGCOMM Workshop Hot Topics Netw., 2010, pp. 1–6.

I. Mahmud. (Jun, 2020). Coupled MPBBR (C-MPBBR)—Linux Kernel Code. [Online]. Available: https://github.com/imtiaztee/C-MPBBR

N. Leah. (Jun, 2010). FQ-Job Queue Log Viewer. [Online]. Available: http://manpages.ubuntu.com/manpages/bionic/en/man1/fq.1.html

S. Hemminger. (Jun. 2010). NetEm—Network Emulator. [Online]. Available: http://manpages.ubuntu.com/manpages/bionic/man1/netem.8.html

D. Miller. (Jun. 2010). Ehttool-Query or Control Network Driver and Hardware Settings. [Online]. Available: http://manpages.ubuntu.com/manpages/bionic/man8/ehttool.8.html

J. Dugan, S. Elliott, B. A. Mah, J. Poskanzer, and K. Prabhu. (Jun. 2010). iperf3—Perform Network Throughput Tests. [Online]. Available: http://manpages.ubuntu.com/manpages/bionic/en/man1/iperf3.1.html

G. Roualland. (Jun. 10). ifstat - Report InterFace STATistics. [Online]. Available: https://manpages.de.net/man/1/ifstat

S. Hemminger and X. Ding. (Jun. 2010). TCPProbe—Observe the TCP Flow With Kprobes. [Online]. Available: https://github.com/fengidri/tcpprobe/blob/master/tcp_probe.c

C. Paasch, R. Khalili, and O. Bonaventure, “On the benefits of applying experimental design to improve multipath TCP,” in Proc. 9th ACM Conf. Emerg. Netw. Exp. Technol. (CoNEXT), 2013, pp. 393–398.

R. K. Jain, D.-M. W. Chiu, and W. R. Hawe, A Quantitative Measure of Fairness and Discrimination, Eastern Res. Lab., Digip. Equip. Corp., Hudson, MA, USA, 1984.

M. Dianati, X. Shen, and S. Naik, “A new fairness index for radio resource allocation in wireless networks,” in Proc. IEEE Wireless Commun. Netw. Conf., Mar. 2005, pp. 712–717.

IMTIAZ MAHMUD received the B.Sc. degree in computer science and engineering from Hajee Mohammad Danesh Science and Technology University, Bangladesh, in 2015, and the M.S. degree from Kyungpook National University, in 2019, where he is currently pursuing the Ph.D. degree in electronics engineering with the Telecommunication and Network Laboratory. Her current research interests include multipath TCP, congestion control algorithms, and UAVs path planning and network constraints.

TABASSUM LUBNA received the B.Sc. degree in computer science and engineering from Hajee Mohammad Danesh Science and Technology University, Bangladesh, in 2015, and the M.S. degree from Kyungpook National University, in 2019, where she is currently pursuing the Ph.D. degree in electronics engineering with the Telecommunication and Network Laboratory. Her current research interests include MPTCP congestion control algorithms.

YEONG-JUN SONG received the B.S. degree in electronics engineering from Kyungpook National University, in 2019, where he is currently pursuing the M.S. degree in electronics engineering with the Telecommunication and Network Laboratory. His current research interests include TCP congestion control and artificial intelligence in networks.

YOU-ZE CHO (Senior Member, IEEE) received the B.S. degree in electronics engineering from Seoul National University, South Korea, in 1982, and the M.S. and Ph.D. degrees in electrical engineering from the Korea Advanced Institute of Science and Technology, in 1985 and 1988, respectively. Since 1989, he has been with Kyungpook National University (KNU), South Korea, where he is currently a Professor with the School of Electronics Engineering and the Dean of the College of IT Engineering. From August 1992 to January 1994, he worked at the University of Toronto, Canada, as a Visiting Researcher. From March 2002 to February 2003, he worked at the National Institute of Standards and Technology, USA, as a Guest Researcher. In 2017, he served as the President of the Korean Institute of Communications and Information Sciences (KICS). His research interests include mobility management and traffic engineering for wireless and mobile networks, and the transport protocols for future Internet.