BBR-Based Congestion Control and Packet Scheduling for Bottleneck Fairness Considered Multipath TCP in Heterogeneous Wireless Networks

Wenjia Wei, *Member, IEEE*, Kaiping Xue, *Senior Member, IEEE*, Jiangping Han, Yitao Xing, *Graduate Student Member, IEEE*, David S. L. Wei, *Senior Member, IEEE*, and Peilin Hong

***Abstract*—Aiming at improving the performance of Multipath TCP (MPTCP) in heterogeneous wireless network environments, in this paper, by taking the advantages Bottleneck Bandwidth and Round-trip propagation time (BBR) and considering bottleneck fairness, we propose BBR-based Congestion Control and Packet Scheduling scheme, called BCCPS. The proposed BCCPS first considers a BBR-based congestion control algorithm, MPTCP-BBR, for MPTCP by adaptively adjusting the sending rate of each subflow according to the real probing rate rather than the loss information to enhance the goodput while keeping bottleneck fairness with regular TCP. Then, considering that the different sending rates of the subflows in heterogeneous wireless networks will result in the problem of out-of-order (OFO) delivery, a fine-grained packet scheduling scheme is proposed to keep in-order delivery and so as to reduce the application layer completion time. The performance of the proposed scheme is evaluated via both NS-3 and real network scenarios. Experimental results show that our proposed scheme far outperforms existing MPTCP schemes in heterogeneous wireless environment.**

***Index Terms*—BBR, bottleneck fairness, congestion control, multipath TCP, packet scheduling.**

I. Introduction

NOWADAYS, it is common that one communication device has more than one network interface (e.g., a mobile phone with 3 G/4 G and wifi interfaces). However, more than 90% of Internet traffic is transported via TCP [1], while the traditional TCP can only utilize one interface between the two end hosts even though the end hosts are equipped with multiple interfaces. As the available capacity in a single network connection cannot satisfy the goodput and delay demands of high-quality applications, new research trends have moved towards integrating the multiple access technology to gain bandwidth aggregation, such as the work in [2]–[6].

As a new bandwidth aggregation technology at the transport layer, Multipath TCP (MPTCP) [7] has got much attention due to its support for concurrent use of multiple interfaces and compatibility with the existing applications [8]. MPTCP uses multiple subflows for parallel transmission to improve goodput and robustness [9], [10]. However, simultaneous use of multiple subflows may create multiple times of aggression on a single-path TCP flow at the shared bottleneck, which means that the existing congestion control algorithms cannot be directly ported to MPTCP [11]. To address this problem, some coupled congestion algorithms, such as LIA [12], OLIA [13], and BALIA [14] have been proposed to enhance the goodput performance and fairness of MPTCP. These packet-loss-based congestion control schemes often incorrectly interpret a packet loss as the signal of congestion. Such congestion control strategy frequently results in goodput fluctuations in lossy networks, and significantly increases the end-to-end delay even when the loss rate is low [15], [16]. Furthermore, all these schemes simply couple the throughput of all subflows to achieve network fairness, making the overall throughput of an MPTCP connection be no more than that of a single-path TCP on the best end-to-end path. Although this operation can guarantee the fairness of coupled MPTCP subflows in one MPTCP connection and other TCP/MPTCP flows at shared-bottleneck links, it limits the performance of the subflows at non-shared-bottleneck links. If we distinguish bottlenecks of different paths for an MPTCP connection, decouple the subflows at different non-shared-bottleneck links, and only couple the subflows at each shared-bottleneck link, the overall throughput can be improved while achieving bottleneck fairness with single-path TCP flows and other MPTCP flows.

Recently, BBR (Bottleneck Bandwidth and Round-trip propagation time) [17], also named TCP-BBR, a rate-based congestion control scheme, is proposed by Google Inc. to improve TCP's transmission performance. This scheme estimates the end-to-end propagation delay and the available link bandwidth at the bottleneck of a subflow to determine the sending rate of the subflow. BBR tries to provide high link utilization while avoiding queueing packets in intermediate routers' buffers. Thus,

Manuscript received October 29, 2020; revised December 23, 2020; accepted December 24, 2020. Date of publication December 29, 2020; date of current version February 12, 2021. This work was supported in part by the National Natural Science Foundation of China under Grant 61972371, and in part by the Youth Innovation Promotion Association of the Chinese Academy of Sciences (CAS) under Grant 2016394. The review of this article was coordinated by Dr. B. Mao. *(Corresponding author: Kaiping Xue.)*

Wenjia Wei, Jiangping Han, and Peilin Hong are with the Department of Electronic Engineering and Information Science, University of Science and Technology of China, Hefei 230027, China (e-mail: wwj2014@mail.ustc.edu.cn; jphang@mail.ustc.edu.cn; plhong@ustc.edu.cn).

Kaiping Xue and Yitao Xing are with the School of Cyber Security, University of Science and Technology of China, Hefei 230027, China (e-mail: kpxue@ustc.edu.cn; ytxing@mail.ustc.edu.cn).

David S. L. Wei is with the Department of Computer and Information Science, Fordham University, Bronx, NY 10458 USA (e-mail: wei@cis.fordham.edu).

Digital Object Identifier 10.1109/TVT.2020.3047877

0018-9545 © 2020 IEEE. Personal use is permitted, but republication/redistribution requires IEEE permission.
See https://www.ieee.org/publications/rights/index.html for more information.

Authorized licensed use limited to: Kongu Engineering College. Downloaded on June 29,2024 at 06:23:07 UTC from IEEE Xplore. Restrictions apply.

BBR can not only improve the goodput but also reduce end-to-end delay of a TCP flow in wireless networks [18], so BBR can be further considered in improving MPTCP's performance in Heterogeneous Wireless Networks. However, if we only implement MPTCP over multiple independent BBR-based TCP subflows without overall consideration about congestion control and packet scheduling of subflows in an MPTCP connection, the transmission performance of MPTCP can not be guaranteed and the fairness with other TCP/MPTCP flows can not be achieved.

In this paper, we first propose a BBR-based coupled congestion control algorithm for MPTCP, named MPTCP-BBR, which integrates the advantages of multipath transmission and BBR. MPTCP-BBR can improve the overall goodput and reduce end-to-end delay while achieving bottleneck fairness with other TCP/MPTCP flows. MPTCP-BBR completes the detection of the shared bottleneck set when each BBR subflow of an MPTCP connnetion periodically probing bandwidth and RTT. Then, MPTCP-BBR couples the growth rate of subflow at each shared bottleneck in Probing Bandwidth state and adaptively adjusts the sending rate of subflows to balance the loads while ensuring fair bandwidth allocation with regular TCP flows and other MPTCP flows. Furthermore, all the existing predictive scheduling schemes, e.g., [19], [20] work on packet-loss-based congestion control, which rely on modeling congestion window (CWND) changes in multiple scheduling cycles when predicting the throughput on each subflow. The sending rate of TCP-BBR does not change according to a specific regular rule, so a new packet scheduling algorithm needs to be designed based on our proposed MPTCP-BBR. Therefore, we further propose a new fine-grained packet scheduling scheme for MPTCP-BBR. Specifically, our packet scheduling scheme works in two phases to accommodate data transmission with different sizes: Initially, it sends data redundantly from multiple paths, which can use more bandwidth consumption in exchange for less delay. This operation is beneficial to mouse flows since they are latency-sensitive. After a specific amount of data is sent out, the scheme switches to a regular MPTCP mode with non-redundant packet scheduling and delivers packets over all end-to-end communication paths with regard to the path quality.

To sum up, our innovative work about BCCPS in this paper mainly includes two parts: a new BBR-based bottleneck fairness considered coupled congestion control algorithm and a fine-grained packet scheduling algorithm for MPTCP used in heterogeneous wireless networks. The first part aims to improve transmission performance while achieving bottleneck fairness. The second part aims to reduce the number of out-of-order packets within multiple asymmetric paths at the receiver. The main contributions of our work can be further summarized as follows:

1) By leveraging BBR, we propose a new bottleneck fairness considered coupled congestion control algorithm, named MPTCP-BBR, which probes the available bottleneck link bandwidth and the propagation delay of each TCP subflow. Based on these probed information, MPTCP-BBR dynamically judges shared-bottleneck sets for all subflows in a MPTCP connection and adaptively adjusts the sending rates on each subflow to balance the congestion and optimize the goodput while achieving bottleneck fairness with other TCP flows and MPTCP flows at shared bottlenecks.
2) In order to reduce the completion time caused by the out-of-order delivery problem, our scheme employs a new fine-grained packet scheduling scheme, which is implemented on the basis of our proposed MPTCP-BBR. Furthermore, we divide the scheduling procedure into two phases: the redundant data transmission phase and the regular MPTCP packet scheduling phase. This design can both achieve high goodput for elephant flows and low completion time for mouse flows.
3) We conduct experiments with NS-3 and in a real network environment. The experimental results show that our proposed scheme can reduce the out-of-order packets in the receiver's buffer by $>50\%$ and increase the goodput by $>30\%$ compared with the comparison schemes, while achieving bottleneck fairness with other TCP/MPTCP flows.

The remainder of this paper is structured as follows. Section II reviews the related work on congestion control and packet scheduling. In Section III, we present the system design of our proposed BCCPS. The performance analysis is given in Section IV, and finally Section V gives the concluding remarks of this work and discusses the future work.

II. Related Work and Background

As an extension to TCP, MPTCP has been developed to meet such expection by taking into account the multi-homing properties of the end-hosts. However, the legacy MPTCP only spreads the data to all available paths to aggregate the bandwidth. In order to enhance the performance of MPTCP in respect of goodput and end-to-end delay, and while achieving fairness. Many studies have been conducted to deal with these challenges by means of congestion control and packet scheduling. Here, we will introduce some representative solutions in these two aspects, and give a brief introduction of BBR.

A. Congestion Control

To meet the aforementioned requirements, an MPTCP congestion control is expected to possess three features [11], namely (1) "Improve Goodput," (2) "Do No Harm," and (3) "Balance Congestion," in which "Do No Harm" means that an MPTCP connection should be TCP friendly which is called *Network Fairness*. Many congestion control schemes based on *Network Fairness*, such as LIA [12], OLIA [13], and BALIA [14], have been proposed. In order to be friendly to TCP, *Network Fairness* restricts that the total goodput of an MPTCP connection in the network is no more than that of a regular TCP, whether or not the subflows share a bottleneck with TCP. Thus, these schemes fail to maximize the goodput of the disjoint paths. On the contrary, *Bottleneck Fairness* only restricts the total goodput of the subflows sharing the same bottleneck not bigger than that of the TCP path sharing this bottleneck.

Besides, the above mentioned congestion control schemes are all packet-loss-based. They always interpret a packet loss event

Authorized licensed use limited to: Kongu Engineering College. Downloaded on June 29,2024 at 06:23:07 UTC from IEEE Xplore. Restrictions apply.

as the signal of congestion. They change the CWND based on the strategy of Additive Increase Multiplicative Decrease (AIMD) and only modify the congestion avoidance phase to perform coupled congestion control for all subflows according to the network condition. When a packet loss is detected at the sender, all of these schemes move traffic away from lossy subflows and halve the CWND. Such strategy results in the frequent throughput fluctuations in lossy networks, especially in wireless networks, with random packet loss caused by link errors. Thus, the performance of MPTCP will be seriously degraded. Besides, the packet-loss-based congestion control also suffers bufferbloat in cellular networks, which causes excessively long delay and bring about no contribution to the goodput improvement [15]. Cao *et al.* [21] proposed a delay-based MPTCP congestion control algorithm, named wVegas, which takes the queuing delay as the congestion signal and balance the traffic among subflows. This algorithm can avoid the increasing of end-to-end delay. However, it also cannot perform well in lossy network environments since a random packet loss event will cause the estimated queuing delay increase, which represents that a congestion happens in the corresponding path, the scheme will set the sending rate of this subflow to a very small value, which will reduce the overall goodput of MPTCP.

Machine learn have been leveraged to improve network performance from several aspects [22]–[24]. Meanwhile, several learning-based congestion control approaches have been proposed recently. Li *et al.* [25] proposed a reinforcement learning-based congestion control scheme for MPTCP, named SmartCC, which learns a set of congestion rules and makes online congestion control decisions adaptively to fit different network situations. Xu *et al.* [26] proposed a deep reinforcement learning (DRL)-based congestion control scheme for MPTCP, named DRL-CC, which dynamically and jointly adjusts all subflows' CWND with a DRL agent. Learning-based approaches have the potential to converge to the best decision to improve goodput under the network conditions that have occurred in offline training stage. However, these approaches, such as SmartCC and DRL-CC, might have a worse performance when facing new network conditions and have a long convergence time to achieve the best sending rate on each subflow [27].

B. Brief Introduction of BBR

BBR [17] is a novel congestion control scheme proposed by Google Inc. to improve the performance of TCP. The objective of BBR is to maximize delivered rate and also minimize end-to-end delay by reducing the queuing delay. Instead of filling the bottleneck buffer, BBR tries to reach the optimal operating point [28] by keeping the amount of inflight packets I in the link to the BDP (Bandwidth Delay Product).

As shown in Fig. 1, BBR implements a four-stage rate adjustment scheme, which includes: **Startup**, **Drain**, **ProbeBW**, and **ProbeRTT**. It adaptively changes the sending rate and takes BDP as its convergence condition. **Startup** is implemented to obtain the link bandwidth as quickly as possible at the beginning of the transmission and **Drain** is the reverse process of **Startup**, which reduce the inflight packets in the link. **ProbeBW** is implemented to estimate the bottleneck bandwidth, and **ProbeRTT** is implemented to probe the minimum round-trip propagation time. BBR estimates the bottleneck bandwidth and RTT periodically and adjusts its sending rate according to the real-time estimated link capacity. This can not only improve the goodput but also reduce the end-to-end delay.

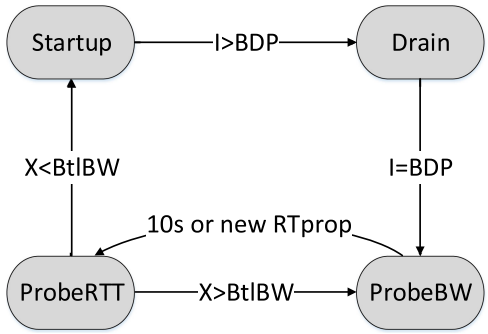


Fig. 1. State-transition diagram of BBR congestion control algorithm.

C. Packet Scheduling

MPTCP establishes a connection with multiple subflows on different paths. When an MPTCP sender has data to send, it must choose a path over which to send that data. Round-Robin (RR) is the simplest scheduling algorithm for MPTCP, in which all subflows have the same priority and the sender just schedules data from the send buffer in sequence to the available send windows of all subflows in polling order. Unfortunately, this simple means cannot alleviate the effects brought up by heterogeneous path characteristics, and it causes out-of-order delivery, where packets with larger sequence number may arrive at receiver earlier than the packets with smaller sequence number, and have to wait until the arriving of the packets with smaller sequence numbers. In current MPTCP specification [7], Lowest RTT First (LRF) [29] is used as the default scheduling algorithm, which relies on the RTT measured on each subflow, and preferentially sends data over the subflow having the lowest RTT. However, LRF also fails to guarantee packets' arriving at the receiver in order. In-order delivery among different paths remains a main challenge for multipath transmission, where out-of-order packets will cause the head-of-line blocking problem at receiver [30].

Moreover, this problem will become more serious in the scenarios with asymmetric communications, where multiple paths have significantly different end-to-end delay. Firstly, the receiver has to maintain a great number of out-of-order packets for reordering, which degrades transmission efficiency. At the same time, in multihomed wireless mobile networks, a mobile device generally has limited memory capacity, and thus has small free space for the receive buffer. Furthermore, the increased out-of-order data and selective acknowledgments will result in more unnecessary fast retransmissions.

Quite a few intelligent predictive packet scheduling algorithms have been proposed to minimize the number of out-of-order packets, [19] and [20], in which the amount of data

Authorized licensed use limited to: Kongu Engineering College. Downloaded on June 29,2024 at 06:23:07 UTC from IEEE Xplore. Restrictions apply.

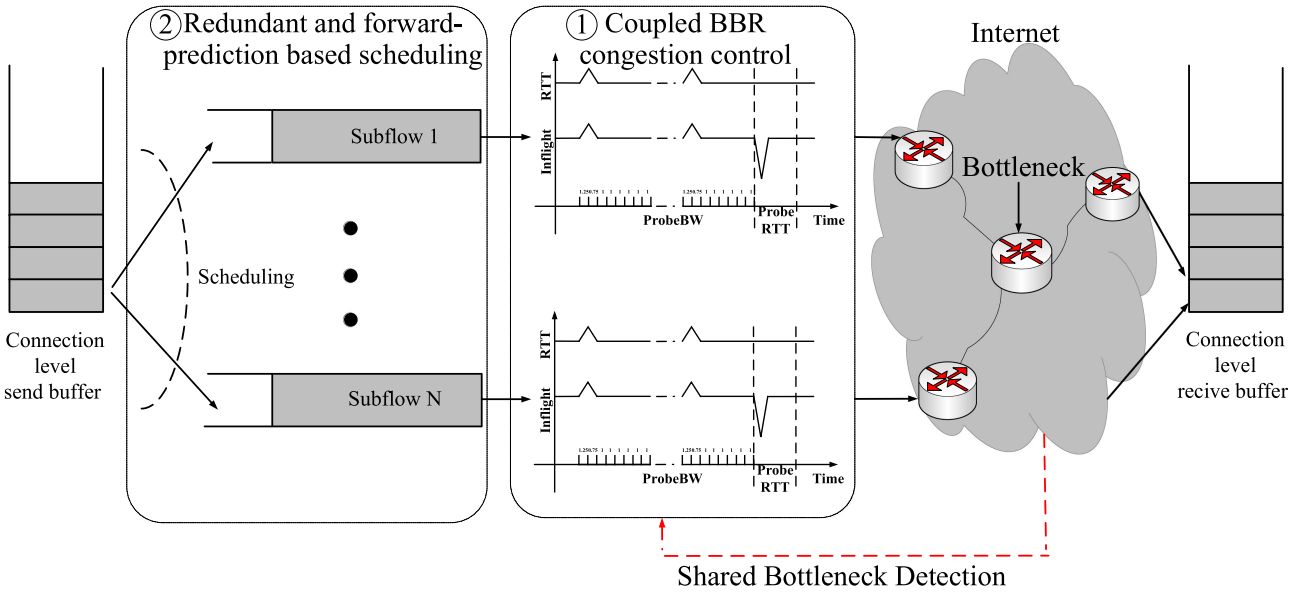


Fig. 2. Framework of BCCPS, which consists of a well-designed BBR-based congestion control algorithm and a fine-grained prediction-based packet scheduling scheme. ①BCCPS groups the subflows into different shared-bottleneck sets according to the state of each subflow. The subflows share the same bottleneck are implemented with coupled congestion control while subflows that are not in a same shared-bottleneck set act as single BBR flows. ②BCCPS then estimates the number of packets scheduled to each subflow based on each subflow's BtlBW and RTT.

scheduled on each subflow is in proportion to the estimated bandwidth of the path. Delay-aware Packet Scheduling (DAPS) [19] takes the network latency into consideration at the sender, which dispatches packets to multiple schedulers instead of the paths in circular order. Compared with DAPS, BLocking ESTimation (BLEST) [20] takes into account the variation characteristics of the fast subflow's CWND, which further improves the accuracy of the estimation of the number of packets transmitted by the fast subflow. However, BLEST does not consider the impact of packet loss on the CWND, and thus it's unable to accurately estimate the CWND when the packet loss occurs. DEcoupled Multipath Scheduler (DEMS) [31] aims to reduce the download time through forward and backward bidirectional scheduling. DEMS is designed for the transmission of small files and requires that the size of application data cannot exceed its MPTCP layer sending buffer. When the transmitting end of the file is large, for example, the data size exceeds the length of the sending buffer of the MPTCP layer, the performance of DEMS degrades severely [32]. Reinforcement Learning based Scheduler (ReLeS) leverages DRL techniques to learn a neural network through experience and generates generate the control policy for packet scheduling to improve goodput, shorten the completion time and reduce the out-of-order queue [32]. However, ReLeS might schedule too many packets on a poor subflow at the very beginning when facing new network conditions. This will have a great influence on the completion time of mouse flows.

Meanwhile, all the proposed predictive scheduling schemes, e.g., BLEST, work on packet-loss-based congestion control, which relies on modeling CWND changes in multiple scheduling cycles when predicting the throughput on each subflow. The send rate of BBR congestion control algorithm does not change according to specific regular rule, these scheduling algorithms cannot work well on a BBR-based congestion control algorithm.

III. System Design of Our Proposed Scheme

The system overview of the proposed BCCPS scheme is illuminated in Fig. 2. Assume that all the data are transmitted through an MPTCP connection with multiple TCP-BBR subflows between the sender and the receiver. The key working components at the sender include a BBR-based coupled congestion control algorithm and a fine-grained packet scheduling algorithm. The latter one is implemented based on implementing the former one.

1) The congestion control scheme consists of a shared bottleneck detection scheme and a coupled congestion control algorithm which can make an MPTCP flow achieve bottleneck fairness with other BBR-based TCP/MPTCP flows. Firstly, based on the detected path parameters of each subflow, the bottleneck shared by two or more subflows can be judged accurately. In BBR's ProbeBW state, If the increase in RTT occurs in two or more subflows of an MPTCP connection simultaneously in a same time, we can preliminarily judge that these subflows share a same bottleneck. The preliminary judgement result of shared bottleneck sets should be further confirmed in the subsequent ProbeRTT state. Then, the proposed coupled congestion control algorithm is implemented to couple the CWND of subflows within the same bottleneck set and adjust the sending rate of each subflow according to the path quality so as to achieve *Bottleneck Fairness*. On the contrary, subflows with different shared-bottleneck sets should be decoupled to achieve a better transmission capacity.
2) The packet scheduling algorithm is implemented based on the implementation result of our congestion control algorithm, which is further divided in two phases: Initially,

Authorized licensed use limited to: Kongu Engineering College. Downloaded on June 29,2024 at 06:23:07 UTC from IEEE Xplore. Restrictions apply.

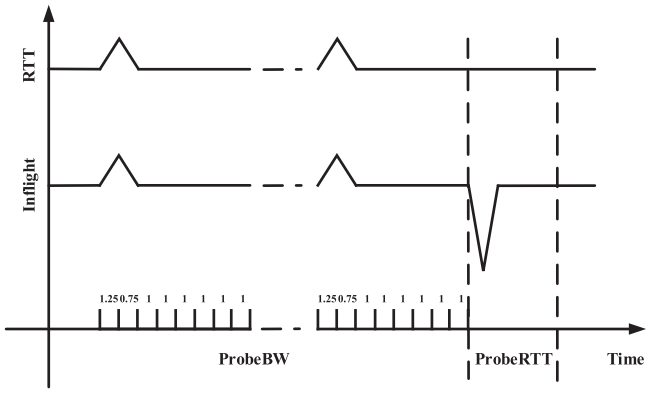


Fig. 3. Inflight packets and RTT in TCP-BBR's four states.

redundant transmission is realized on different subflows, which can shorten the completion time of latency-sensitive mouse flows through more bandwidth consumption. After reaching a fixed statistical threshold, a fine-grained packet scheduling algorithm is run to accurately estimate the completion time of each packet on each subflow based on the values of BtlBW and RTT obtained by the proposed congestion control algorithm.

A. Bottleneck Fairness Considered Coupled Congestion Control

When we introduce BBR into MPTCP, we have to redesign the congestion control scheme such that the developed congestion control possesses all of the three features, namely (i) "Improve Goodput," (ii) "Do No Harm," and (iii) "Balance Congestion" [11]. Therefore, in order to achieve bottleneck fairness, all subflows in an MPTCP connection can be divided into one or more shared-bottleneck sets. In each shared-bottleneck set, MPTCP-BBR should perform equally well as a single-path TCP-BBR flow running on the best path at the shared-bottleneck. Therefore, the overall MPTCP connection can achieve at least as well as the sum of single-path TCP-BBR flows at all shared bottlenecks. Besides, the multipath BBR-based coupled congestion control should be able to promptly move traffic from its most congested paths to other paths as much as possible. For the sake of description, we name our proposed BBR-based congestion control scheme as MPTCP-BBR, and we name TCP with BBR as TCP-BBR.

In order to make the designed scheme to achieve the above mentioned three features simultaneously, we first need to be able to detect all shared-bottleneck sets among subflows in an MPTCP connection, then coupled subflows in each shared-bottleneck set to be fair with a single-path TCP-BBR flow from the same bottleneck. To be noted, a subflow not in any shared-bottleneck set can be treated as a single-path TCP-BBR flow.

1) Network State Collection and Shared Bottleneck Set Detection: As stated in [17], BDP is regarded as the convergence condition of BBR, i.e., the design of BBR aims to make the number of inflight packets converge to BDP. Fig. 3 shows changing trend of inflight packets and RTTs in ProbeBW and ProbeRTT. In the ProbeBW state, BBR adjusts the sending rate with the pacing gain $G \in [1.25, 0.75, 1, 1, 1, 1, 1, 1]$ in an eight-phase cycle to detect the maximum BtlBW. Consequently, BBR periodically spends an RTprop interval at a pacing gain "> 1," which increases inflight packets so as to probe for a bigger sending rate. If the number of inflight packets has converged to BDP, some transmitted packets will queue at the bottleneck and not only the RTT of the probed flow but also the RTTs of other flows from the shared bottleneck will increase simultaneously. During the next RTprop, this queue will be removed by set the sending rate at a compensating pacing gain "< 1".

So we can find that in the above process, RTT's increment process shows a linear feature and the subflows traversing the shared bottleneck would be observed with a similar RTT's changing trend. Based on this observation and inspired by the work in [33] and [34], we leverage the feature of RTT's changing trend of different subflows in a shared bottleneck set in ProbeBW to determine whether the subflows share bottleneck. In this way, when the RTTs of two subflows increase nearly at the same time (we set the judgement period as one RTprop), the subflows can be judged as sharing a bottleneck. This detection can be implemented on sender's side.

Due to the influence of background flows in the network, RTT's measurement error will affect the detection accuracy of shared-bottleneck sets. So the previous detection result need to be verified. We introduce a further confirmation mechanism in our scheme to confirm the detected shared-bottleneck set in the subsequent ProbeRTT phase. In order to measure $RTT_{\min}$, BBR uses a periodically execution phase, called ProbeRTT. As defined in BBR, this phase is entered when the value of $RTT_{\min}$ hasn't been updated with a lower measured value for several seconds (default: 10 s). In ProbeRTT, the sender limits its inflight data amount to 4 packets a maximum value. This state lasts for 200 ms and then returns to the previous state. When a subflow drains the bottleneck queue by reducing the sending rate so as to obtain the $RTT_{\min}$. This will promote other subflows in the same bottleneck to detect the new RTprop and further enter ProbeRTT together. Thus, subflows traversing from a same bottleneck could approximately synchronize the above process of state change at the same time. When different subflows enter the ProbeRTT state nearly at the same time, we can confirm the subflows judged to share a bottleneck previously actually share the bottleneck.

Specifically, our shared-bottleneck set detection mechanism works as follows:

- S1: The MPTCP sender monitors the RTTs of all subflows in respective ProbeBW states. When the RTT of one subflow increases, the sender further detect whether there are other subflows whose RRTs are also linearly increasing in the same time. If so, these subflows can be judged to share a bottleneck, which can be treated to belong to a shared-bottleneck set.
- S2: Furthermore, for a shared bottleneck set judged previously, if the minimum RTTs of the subflows in a shared bottleneck set are measured simultaneously in the ProbeRTT state, then this bottleneck set can be confirmed finally and all the subflows in this set should be coupled together. Otherwise, the previously judged shared-bottleneck set should be cancelled.

Authorized licensed use limited to: Kongu Engineering College. Downloaded on June 29,2024 at 06:23:07 UTC from IEEE Xplore. Restrictions apply.

This procedure with two steps needs to ensure that all subflows are determined, each of which belongs to a shared bottleneck set or is independent and doesn't share a bottleneck with any other subflows. What is more, this procedure should be executed periodically and the detected shared bottleneck sets can be dynamically changed as on each subflow, as BBR probes the link state periodically and the probe result is always dynamic changed.

2) Bottleneck Fairness Considered Coupled Congestion Control: [18], [35] In our proposed congestion control mechanism, based on the result of shared bottleneck set judgement, the subflows in a shared bottleneck set are coupled together, and the subflows that are not in the same shared-bottleneck set should be decoupled and perform congestion control processes independently of each other, which represents *Bottleneck Fairness*. Besides, we also need to ensure fairness between different subflows with different RTTs that are coupled together from a shared bottleneck link. The problem of the fairness about TCP-BBR flows with different RTTs has been discussed in some literatures, e.g., [36]–[39], in which long RTT flows always occupy more bandwidth than short RTT flows at the shared bottleneck link. The cause of this phenomenon is that, a long RTT flow floods in a larger volume of excess inflight packets than a short RTT flow does during the probing period (one RTprop), which dominating the queue backlog as well as the sending rate. Therefore, this factor needs to be considered in the design of our fairness considered scheme.

In order to achieve feature (i), i.e., "Improve Goodput," MPTCP should perform equally well as a single-path TCP running on the best path from a bottleneck, in our scheme, which means MPTCP-BBR should spend as much time as that spent by a single-path TCP-BBR in the probing period. According to [36], the time length of the probing period for all the subflows in a shared-bottleneck set S_i can be chosen as defined in Eq. (1):

$$RTT' = \max_{r \in S_i}\{RTprop_r\}, \tag{1}$$

where $RTprop_r$ is the minimum RTT estimated by subflow r. RTT' is the maximum $RTprop_r$ belongs to the MPTCP connection in the shared-bottleneck set S_i. The computation of RTT' guarantees that all subflows in the same shared-bottleneck set probe respective sending rate in each subflow will be proportional to its pacing gain. Let $x_r(t)$ be the measured sending rate of subflow r at time t, the rate control in can be expressed as:

$$BDP_r = \max\{x_r(t)\} \times RTT' \qquad t \in [T-W, T], \tag{2}$$

$$R_r = \begin{cases} G_r \times \max\{x_r(t)\} & I \leq BDP_r, \\ 0 & I > BDP_r. \end{cases} \tag{3}$$

where $\max\{x(t)\}$ is the maximum delivery rate during the last period W which is typically 6 to 10 times of $RTprop$.

In ProbeBW state, G_r represents the gain of sending rate in one of the eight-phase on subflow r. A single-path TCP-BBR controls the pacing rate as $G \in [1.25, 0.75, 1, 1, 1, 1, 1, 1]$. When multiple subflows are competing with each other on a shared-bottleneck link, the bandwidth obtained by each subflow should be proportional to its occupied buffer size in the link queue. Thus, it is reasonable for a subflow to set the parameter α_r as a knob to control the aggressiveness of competition for bandwidth consumption. Then each subflow's pacing gain can be set as:

$$G_r \in [1.25, 0.75, \alpha_r, \alpha_r, \alpha_r, \alpha_r, \alpha_r, \alpha_r]. \tag{4}$$

Moreover, in order to have feature (ii), i.e., "Do No Harm" and feature (iii), i.e., "Balance Congestion," the throughput of each subflow in a shared-bottleneck should be equal to the throughput of the best subflow with $\max BtlBW_r$ from the same bottleneck link. Then, the throughput of each subflow is allocated proportionally according to the currently measured bandwidth. Therefore, the parameter α_r can be computed as in Eq. (6).

$$BtlBW_r = max\{x_r(t)\} \qquad t \in [T-W, T], \tag{5}$$

$$\alpha_r = \frac{\max\limits_{r \in S_i} BtlBW_r}{\sum_{r \in S_i} BtlBW_r}. \tag{6}$$

In this way, a new BBR-based coupled congestion control algorithm, i.e., MPTCP-BBR, is developed for MPTCP. MPTCP-BBR possesses the three before-discussed features. Algorithm 1 gives pseudocode of our proposed MPTCP-BBR.

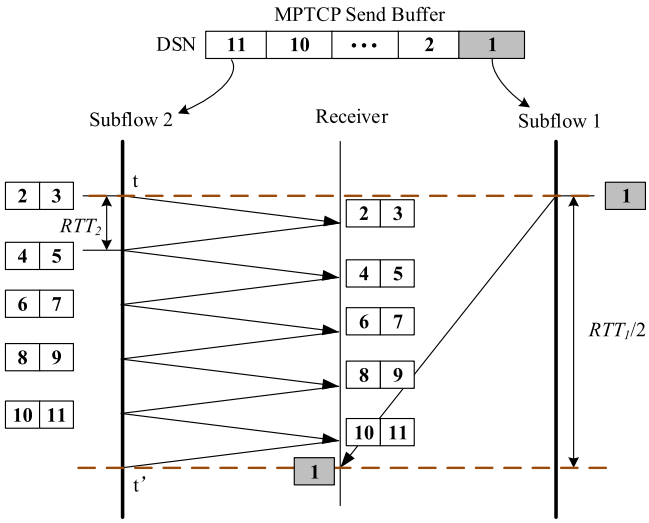


Fig. 4. Example of a HoL-blocking state. Packet 1, delivered on the slow path delays the entire transmission, even though packets 2-11 have already arrived.

B. Fine-Grained Packet Scheduling

In this subsection, based on the path quality detected by BBR, we further introduce a new scheduling algorithm that addresses out-of-order delivery problem and spurious retransmissions to increase application performance in heterogeneous scenarios.

In heterogeneous scenarios, due to the uneven performance between different subflows, the data allocated to different subflows will not reach the receiver at the same time, causing out-of-order problems. The performance of MPTCP will seriously degraded as the subflow performance difference increases, especially for mouse flows [40]. As shown in Fig. 4, a first packet is sent on a slow path, and subsequent packets are sent on the fast path to fill the receive buffer. While the subsequent packets

Authorized licensed use limited to: Kongu Engineering College. Downloaded on June 29,2024 at 06:23:07 UTC from IEEE Xplore. Restrictions apply.

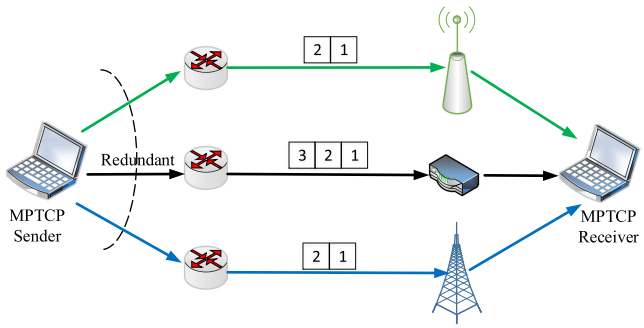


Fig. 5. In the initial phase, we use redundant packet transmission to reduce latency and jitter.

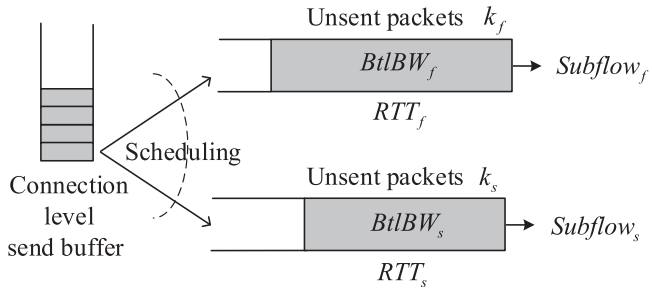


Fig. 6. Forward-prediction based scheduling to reduce latency.

are received in a timely manner, the transmission is blocked, waiting for the first packet to be received.

Algorithm 1: Bottleneck Fairness Considered Coupled Congestion Control Algorithm.

For an MPTCP connection, judge shared-bottleneck sets in ProbeBW and ProbeRTT states and count the number n_i of subflows in each shared-bottleneck set i
for each shared bottleneck set S_i **do**
  Initialize $\alpha_r = 1/n_i$,
  for each subflow r **do**
    if the subflow r in the shared-bottleneck set **then**
      RTT' = get_current_maxRTT();
      $BDP_r = max\{x_r(t)\} \times RTT' \quad t \in [T - W, T]$;
      $\alpha_r = \frac{\max BtlBW_r}{\sum_{r \in S} BtlBW_r}$;
      $G_r \in [1.25, 0.75, \alpha_r, \alpha_r, \alpha_r, \alpha_r, \alpha_r, \alpha_r]$;
      if *Infight packets* $\leq BDP_r$ **then**
        $R_r = G_r \times \max\{x_r(t)\}$;
      else if *Infight packets* $> BDP_r$ **then**
        $R_r = 0$;
      end
  end
end
for each of other independent subflows, r , which doesn't belong to any shared-bottleneck set **do**
  r *acts as a single-path TCP-BBR flow*;
end

In order to achieve high goodput for elephant flows and low latency for mouse flows, we divide the process into two phases. In the initial phase, packets are redundantly transmitted over multiple paths until the amount of transmitted packets reaches a certain threshold. In the second phase, the parameters of different paths are collected, and the sender allocates packets over multiple paths according to the path quality.

1) Redundancy Packet Transmission: In initial phase, the sender sends data packets redundantly over multiple paths in heterogeneous network. As shown in Fig. 5, although WiFi, ethernet, and LTE have different link bandwidth, delays, and packet loss rates, BCCPS sends the same packet sequence on different subflows, exchanging bandwidth consumption for reducing latency. This approach guarantees the lowest possible latency in existing best-effort networks. It is essential to mouse flows, since retransmissions due to packet drops can increase the end-to-end delay obviously. We heuristically mandate that flows less than or equal to 100 KB are considered mouse flows, and are replicated to achieve better latency. This threshold value is chosen in accordance with many existing literatures [41]–[43]. This threshold is small, so it will not waste too much bandwidth. At the same time, it can also collect the network parameters (i.e., BtlBW and RTT) for the second phase with BBR's pacing mechanism.

2) Forward Prediction Packet Scheduling: In this phase the packets generated by applications come into the send buffer. The algorithm allocates packets to different subflow according to the completion time to solve the performance degradation problem with path heterogeneity.

This algorithm aims at scheduling more packets on a subflow than what it can currently send, so the queues may therefore build up at the sender. For each packet in the queue, scheduler selects the appropriate subflow to transmit according to its arrival time in FIFO order. As shown in Fig. 6, assume that there are packets needed to be allocated to $Subflow_f$ and $Subflow_s$. If the faster subflow $Subflow_f$ in terms of RTT has available capability, current scheduled packets can simply be dispatched to $Subflow_f$. As the unsent packets on $Subflow_f$ increases, the time that new packets wait to be dispatched to $Subflow_f$ will increase. If $Subflow_f$ does not have available space and subsequent packets can arrive faster via $Subflow_s$, then these packets should be scheduled to and sent via $Subflow_s$.

When scheduling a connection level packet, the algorithm estimates its arrival time, denoted by T_r, if sent over subflow r, which is the end-to-end delay of subflow r. Then it chooses the subflow with the earliest arrival time. T_r is the sum of two major delay: the delay introduced in network T^r_{net} and the delay introduced in sendbuffer T^r_{wait}.

$$T_r = T^r_{net} + T^r_{wait}, \tag{7}$$

$$T^r_{net} = \frac{RTT_r}{2} + q_r \cdot RTT_r, \tag{8}$$

$$T^r_{wait} = \frac{k_r}{BtlBW_r}, \tag{9}$$

Authorized licensed use limited to: Kongu Engineering College. Downloaded on June 29,2024 at 06:23:07 UTC from IEEE Xplore. Restrictions apply.

Algorithm 2: Packets Scheduling Algorithm.

Input: $BtlBW_r$, RTT_r and loss rate q_r of subflow r;
Initialize initial *count_data* = 0,
`/* The initial phase */`
for count_data $< T_{\text{threshold}}$ **do**
  count_data ++ ;
  redundantly schedule the current packets to
  all the available subflows;
end
`/* The second phase */`
for each connection level packet **do**
  for *each available subflow* r **do**
    $T^r_{net} = \frac{RTT_r}{2} + q_r \cdot RTT_r$;
    $T^r_{wait} = \frac{k_r}{BtlBW_r}$;
    $T_r = T^r_{net} + T^r_{wait}$;
    if $T_r < T_{min}$ **then**
      $T_{min} = T_r$;
      $sch\_sf = r$;
    end
  end
  schedule the packet to subflow sch_sf.
end

where T^r_{net} represents the time it takes for the data to arrive at the receiver in the network transmission with the packet loss rate q_r. When the sender detects a loss event, it updates the loss rate q_r, calculates the average loss rate, and forecast the time arriving at the destination. In the calculation process of T^r_{net}, we assume that the lost packet can successfully arrive at the receiver by one retransmission. T^r_{wait} is related to the number of packets scheduled on this subflow. In order to make the data on different subflows reach the receive buffer at the same time, the fast subflow will be scheduled for much more data, so the waiting time on the fast subflow needs to be estimated. The estimation is performed based on a subflow's $BtlBW_r$, and the number of unsent packets in queue k_r.

When the scheduling of a packet is completed, the packet will be sent via the allocated subflow. If the queue is empty of packets on this subflow, the current scheduled packet can be sent on the subflow immediately. Otherwise, the packet have to wait in the queue and will be sent by the assigned subflow later when the previous scheduled packets in the queue are sent out. In this way, all these packets will arrive at receiver in order. Algorithm 2 give the pseudocode about how the packet is scheduled, based on the RTT, BtlBW, and the loss rate in our proposed BCCPS.

IV. Performance Analysis

In this section, we first evaluate our BCCPS proposed in this paper through NS-3 simulator [44]. The MPTCP NS-3 code is provided by Google MPTCP group [45]. Then we implement BCCPS in Linux kernel base on MPTCP v0.94 and test it in real network. We first evaluate the performance of BCCPS's congestion control algorithm MPTCP-BBR, and then evaluate the performance of our packet scheduling algorithm based on MPTCP-BBR. For the performance comparison in the aspect of congestion control, we compare our scheme with MPTCP using LIA and BALIA. Furthermore, for the performance comparison in the aspect of packet scheduling, RR and LRF and BLEST [20] are used to compare with our scheme.

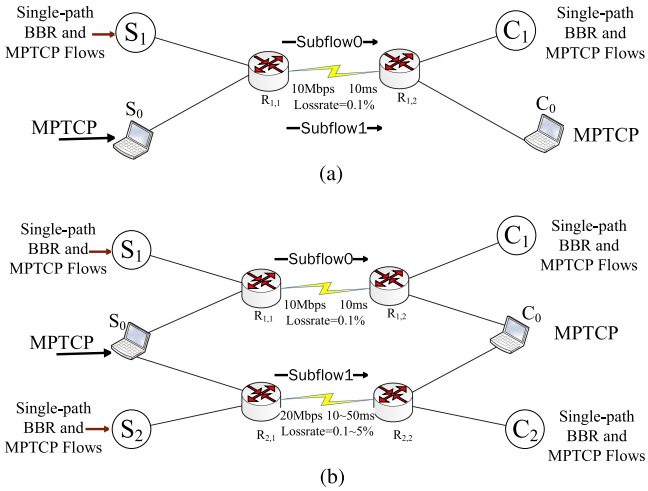


Fig. 7. Base topology for BCCPS evalution. (a) Shared bottleneck scenario. (b) Non-shared bottleneck scenario.

TABLE I
Network Characteristics in NS3

	Shared bottleneck	Non-shared bottleneck
Capacity [Mbps]	10	10-20
Delay [ms]	10-20	10-50
Loss [%]	0.1	0.1-5
Router buffer [BDP]	1	1

Performance Metrics: 1) Goodput: the application-level throughput of a communication, which can accurately reflect algorithm performance, especially for bulk transmission. 2) Average MPTCP OFO queue size: the average out-of-order queue size counted at receiver, which can reveal the effectiveness of scheduling algorithm. 3) Download time: the completion time to download a specific size file, especially for small-size data transmission.

A. Performance Evaluation in NS-3

1) Simulation Setup: We consider two typical scenarios, Shared bottleneck scenario shown in Fig. 7(a) and Non-shared bottleneck scenario shown in Fig. 7(b). The simulation parameters and their values are shown as Table I.

1) Fig. 7(a) is the shared bottleneck scenario. In this scenario, the network has a common bottleneck. Two subflows ($Subflow_0$ and $Subflow_1$) are established between MPTCP client (C_0) and server (S_0). The subflows compete with each other in this bottleneck. Besides, we will introduce single-path BBR and MPTCP flows as background traffic into this bottleneck to analysis the fairness. The bottleneck bandwidth is set to 1 Mbps and the link delay is set to 10 ms. Bandwidth of other links is set to 100 Mbps.
2) Fig. 7(b) is the non-shared bottleneck scenario. In this scenario, two subflows are established between MPTCP

Authorized licensed use limited to: Kongu Engineering College. Downloaded on June 29,2024 at 06:23:07 UTC from IEEE Xplore. Restrictions apply.

client (C_0) and server (S_0), $Subflow_0$ and $Subflow_1$ do not share a bottleneck. $Subflow_0$ has 1 Mbps bandwidth, 10 ms latency and 0.1% loss rate. In this simulation we keep the parameters of $Subflow_0$ unchanged, while the delay of $Subflow_1$ vary from 10 ms to 50 ms and the loss rate vary from 0.1% to 5%. single-path BBR and MPTCP background flows produce background traffic between clients (C_1, C_2) and servers (S_1, S_2), and compete with MPTCP flow at bottleneck.

2) Congestion Control Algorithm: We first compare BCCPS's congestion control algorithm MPTCP-BBR with MPTCP-LIA and MPTCP-BALIA. In this part of the comparison, we simply use LRF as the scheduling algorithm.

In the shared bottleneck scenario as shown in Fig. 7(a), $Subflow_0$ and $Subflow_1$ share the same bottleneck with single-path flow. In this scenario, both *Bottleneck Fairness* and *Network Fairness* principles will limit MPTCP's aggregate goodput similarly with a single-path BBR flow sharing the same bottleneck. Thus in theory, MPTCP-BBR should achieve nearly equivalent goodput to the single-path single-path BBR flow. In Fig. 7(a), we change the delay of subflow 1 from 10 ms to 20 ms and set the random packet loss on both subflow 0.1%.

Fig. 8(a) shows the detection accuracy of our detection mechanism and loss-and-delay-based mechanism, i.e., DWC, in the shared bottleneck scenario. In the lossy network, using DWC, random packet loss will increase the probability of mis-detection, and moreover, the improperly set delay threshold also easily results in shared bottleneck detection errors. As shown in this figure, MPTCP-BBR's detection accuracy is higher than DWC. Then, we evaluate the goodput of MPTCP-BBR, MPTCP-LIA and MPTCP-BALIA, and set the goodput of single-path BBR as the baseline to compute the ratio. Fig. 8(b) shows the result of transmission goodput comparison in the two scenarios with the same end-to-end delay and different delay on the two paths. We can see that MPTCP-BBR outperforms MPTCP-LIA and MPTCP-BALIA obviously as MPTCP-BBR can continuously probes the current transmission capacity and its transmission goodput is closer to theoretical bandwidth. When the RTT of $Subflow_1$ increases, MP-BBR can still achieves higher goodput. Moreover, the performance of MPTCP-BBR in Fig. 8(b) is close to single-path TCP-BBR, which means that MPTCP-BBR detects bottleneck correctly and achieves fairness with regular single-path TCP-BBR flow at the shared bottleneck.

In the Non-shared bottleneck scenario as shown in Fig. 7(b), $Subflow_0$ and $Subflow_1$ share different bottlenecks with single-path flows. $Subflow_0$ shares the same bottleneck with BBR_0, while $Subflow_1$ shares the same bottleneck with BBR_1. We first measure the accuracy of bottleneck detection of our detection mechanism in this scenario. Fig. 9(a) shows the detection accuracy results. Since the background flow is dynamic on the two paths, it is more difficult to judge in this scenario, but our solution is still better than DWC. In this scenario, each subflow of MPTCP-BBR should behave like a single-path BBR flow to achieve bottleneck fairness, and the aggregated goodput of MPTCP-BBR should be similar to BBR_0+BBR_1. We set the goodput of single-path BBR_0+BBR_1 as baseline evaluate the goodput of MPTCP-BBR, MPTCP-LIA and MPTCP-BALIA.

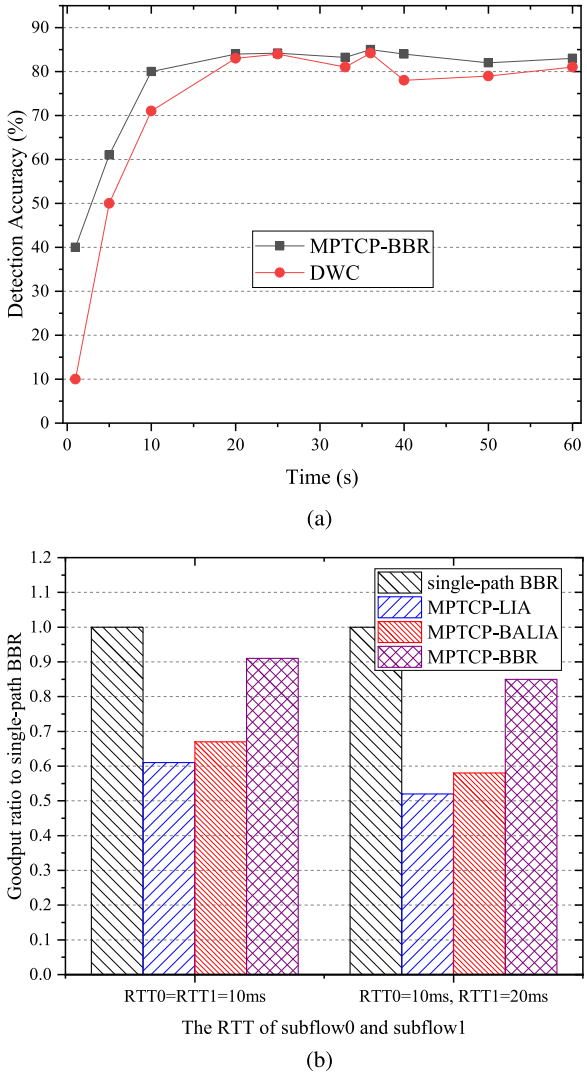


Fig. 8. Bottleneck detection accuracy and goodput in shared bottleneck scenario. (a) Bottleneck detection accuracy. (b) Obtained goodput in shared bottleneck scenario.

In Fig. 9(b), we can see that MPTCP-BBR outperforms MPTCP-LIA and MPTCP-BALIA close to BBR_0+BBR_1, which means that each of the disjoint subflows can reach the goodput of single-path TCP-BBR and a good bottleneck fairness can be achieved.

3) Scheduling Algorithm: We further evaluate the performance improvement brought by the scheduling algorithm in our proposed BCCPS. We compare BCCPS's scheduling algorithm with RR and LRF. In this part of the comparison, BCCPS, MPTCP-BBR-RR and MPTCP-BBR-LRF all use MPTCP-BBR as congestion control algorithm. BLEST, as a predictive scheduling scheme works on loss-based congestion control, which rely on modeling window change in multiple scheduling cycles when predicting the throughput on each subflow, can not be applied to MPTCP-BBR. However, we still compare the scheduling algorithm of BCCPS with MPTCP-BLEST in terms of out-of-order queue size.

We first compare the normalized download time of different schemes (taking the best single-path TCP-BBR as a baseline,

Authorized licensed use limited to: Kongu Engineering College. Downloaded on June 29,2024 at 06:23:07 UTC from IEEE Xplore. Restrictions apply.

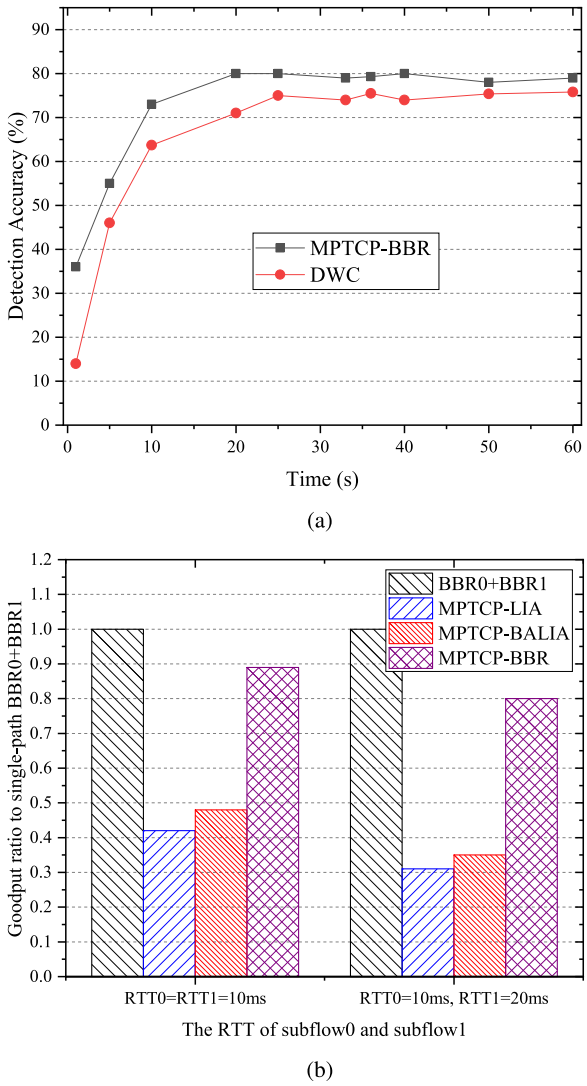


Fig. 9. Bottleneck detection accuracy and goodput in non-shared bottleneck scenario. (a) Bottleneck detection accuracy. (b) Obtained goodput in shared bottleneck scenario.

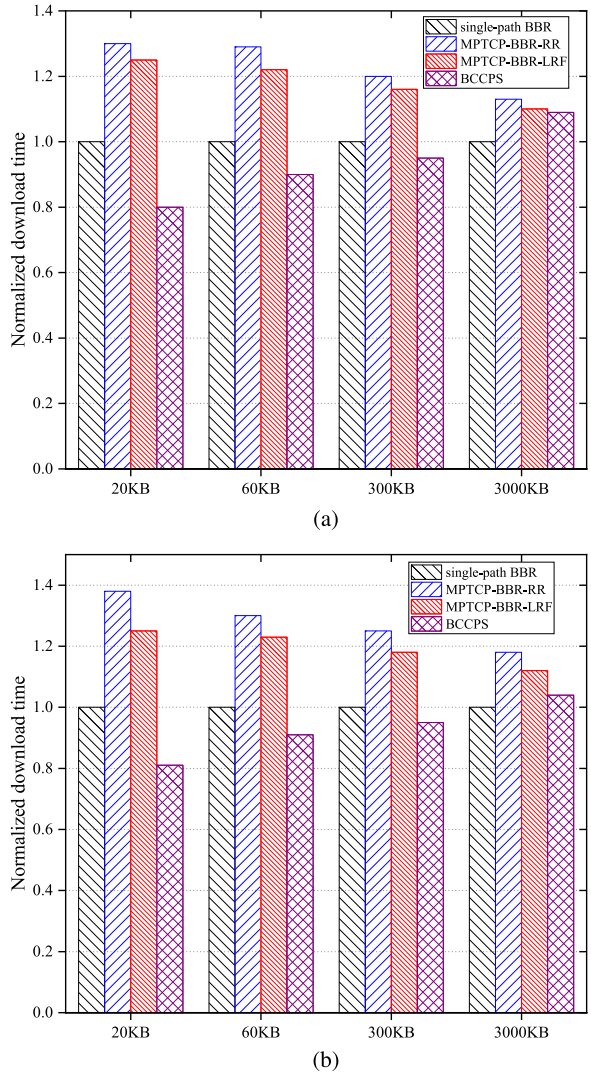


Fig. 10. Normalized download time in shared bottleneck scenario. (a) RTT0 = RTT1 = 10 ms. (b) RTT0 = 10 ms, RTT1 = 20 ms.

which is theoretical performance value) in shared bottleneck scenario. In Fig. 10, we observe that the amount of data and the path heterogeneity have significant influence on the performance of scheduling algorithms. Fig. 10(a) shows that the best single-path BBR outperforms MPTCP-BBR-RR and MPTCP-BBR-LRF when the amount of data is less than 100 KB, since the packet loss has a great impact on the completion time of the mouse flows. Compared with MPTCP-BBR-RR and MPTCP-BBR-LRF, BCCPS avoids the long time caused by packet loss during the startup phase with redundancy packet transmission. When the amount of data is greater than 100 KB, all the three scheduling algorithms perform nearly the same as the best single-path BBR and BCCPS has the highest performance. Fig. 10(b) shows the performance of these scheduling algorithms when the RTTs of subflows are different. BCCPS still performs the best since precisely distributing data packets on both paths yields the optimum regardless of the size of the file.

In non-shared bottleneck scenario, we first exchange a small file with a size of 100 KB between MPTCP client (C_0) and server (S_0) to evaluate the performance of mouse flows in lossy network. Then we conduct a large file transmission to verify the performance of BCCPS in terms of goodput and out-of-order packets.

In Fig. 7(b), we set the loss rate of $Subflow_0$ and $Subflow_1$ to 0.1% and 0.5% respectively. Then we exchange the 100 KB data through the two disjoint paths and they compete with dynamic background traffic. From the Fig. 11, we can see that BCCPS outperforms MPTCP-BBR-RR and MPTCP-BBR-LRF. The result is consistent with the results we measured in shared bottleneck scenario. This proves that BCCPS reduces the completion time of mouse flows since the redundancy packet transmission reduce the possibility of retransmission.

Fig. 12 shows the result of transmission goodput corresponding to different link loss rate when we set the delay of $Subflow_0$ and $Subflow_1$ to 10 ms. Since the RTT settings of the two subflows are equal, we find MPTCP-BBR-RR and

Authorized licensed use limited to: Kongu Engineering College. Downloaded on June 29,2024 at 06:23:07 UTC from IEEE Xplore. Restrictions apply.

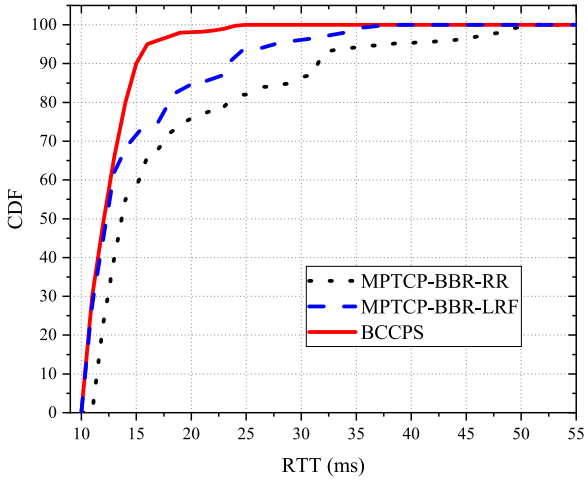


Fig. 11. CDF of end-to-end RTT in non-shared bottleneck scenario.

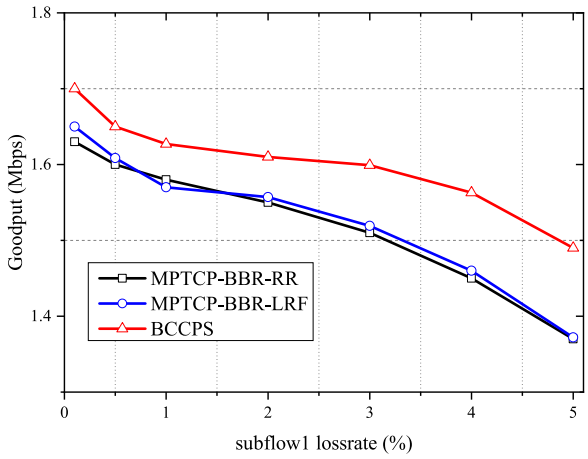


Fig. 12. Obtained goodput with varying loss rate in non-shared bottleneck scenario.

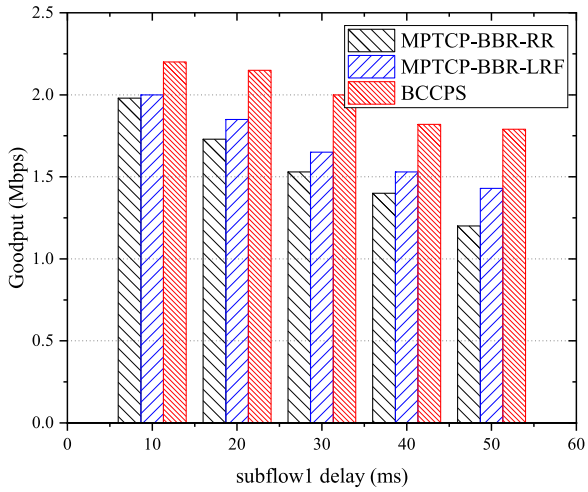


Fig. 13. Obtained goodput with varying link delay of subflow 1 in non-shared bottleneck scenario.

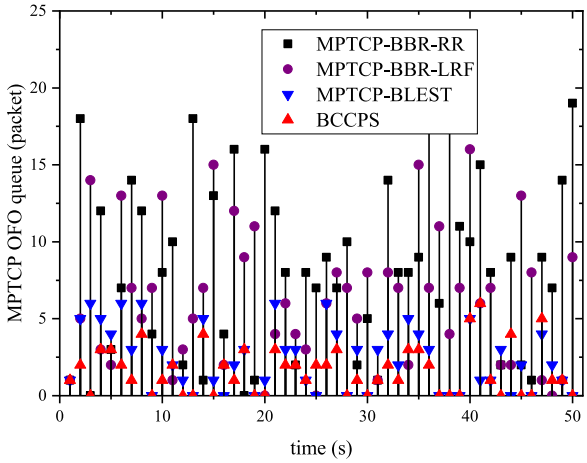


Fig. 14. MPTCP OFO queue in non-shared bottleneck scenario.

MPTCP-BBR-LRF behave nearly the same. We can see that BCCPS performs best regardless of the loss rate. The reason is that BCCPS accurately assigns data to each subflow based on the completion time of each subflow, which maximizes the performance of each subflow and makes its performance better than MPTCP-BBR-RR and MPTCP-BBR-LRF in this scenario.

Fig. 13 shows obtained goodput with varying link delay of $Subflow_1$ in non-shared bottleneck scenario. It is obviously that the goodput of them degrades as the delay increases. Because the increasing delay makes the links more asymmetric. However, we can also see that, with the increase in the distinction of available paths, the superiority of BCCPS over MPTCP-BBR-RR and MPTCP-BBR-LRF becomes more obvious as the scheduling algorithm is able to effectively leverage the path diversity to optimize the aggregate goodput.

As shown in Fig. 14, MPTCP-BBR-RR and MPTCP-BBR-LRF induce more out-of-order packets than MPTCP-BLEST and BCCPS since they do not take the link quality into consideration. MPTCP-BLEST and BCCPS periodically estimate the latest information available in terms of path status and distributes the traffic loads according to the predicted arrival time. Since MPTCP-BLEST does not consider the impact of packet loss, the CWND estimation will be inaccurate when packet loss occurs. However, in this network scenario, BBR can more accurately estimate the network bandwidth, and BCCPS predicts the arrival time of a packet on each subflow based on the accurate bandwidth detected by BBR. Thus, BCCPS allocates packets to different subflows more accurately, reduces the out-of-order queue size and performs better than other reference schemes.

B. Performance Evaluation in Real Network

1) System Settings and Network Parameters: Considering that the existing application server does not support MPTCP, we develop a heterogeneous wireless network test framework based on Socks proxy according to [46], and compare the performance of BCCPS and traditional MPTCP. We implement our BCCPS in the Linux kernel of Socks Proxy and the client. As shown in Fig. 15, we evaluate the performance of MPTCP, MPTCP-BBR, and BCCPS in a constructed non-shared bottleneck scenario as

Authorized licensed use limited to: Kongu Engineering College. Downloaded on June 29,2024 at 06:23:07 UTC from IEEE Xplore. Restrictions apply.

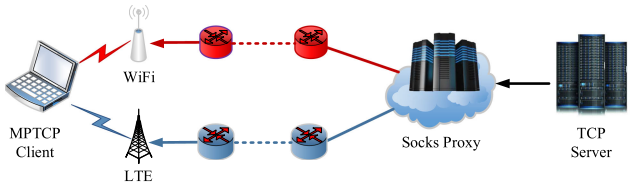


Fig. 15. Real network experiment setup.

TABLE II
REAL NETWORK CHARACTERISTICS

	WiFi	LTE
Capacity [Mbps]	8-14	16-20
Delay [ms]	30-50	80-100
Loss [%]	0.5-1	0.05-0.2

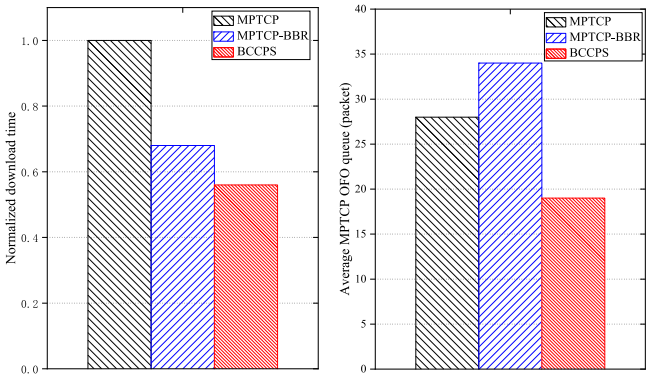


Fig. 16. Normalized download time and average MPTCP OFO queue for bulk traffic in real network experiments.

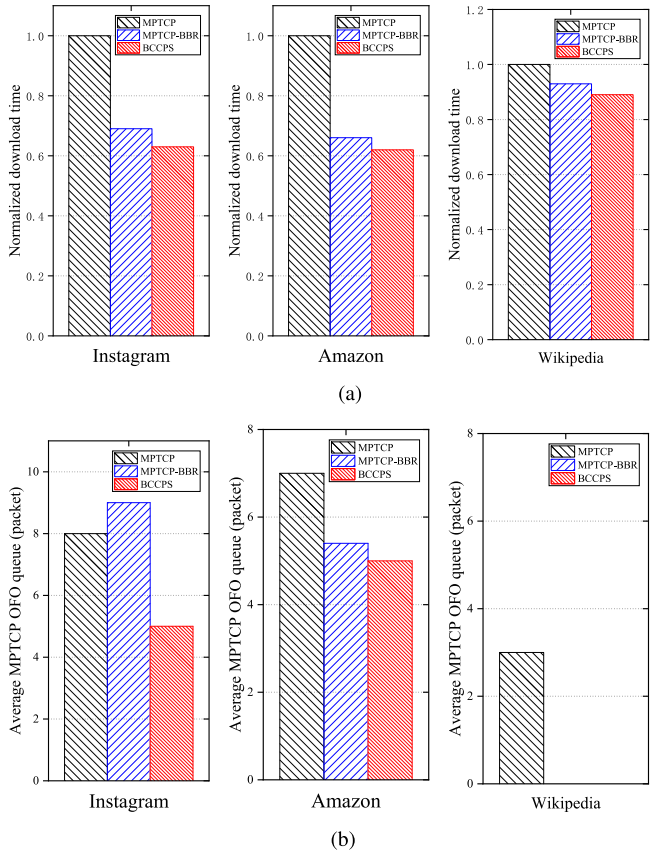


Fig. 17. Normalized download time and average MPTCP OFO queue for web traffic in real network experiments. (a) Normalized download time. (b) Average MPTCP OFO queue.

used in the NS-3 experiments. The client has both WiFi and LTE interfaces, and can downloads data from the application server through WiFi and LTE at the same time. We first measure the single-path network parameters of the WiFi and LTE, then compare the performance of the algorithm against different applications. The network characteristics are summarized as Table II:

2) Bulk Traffic: We first test the performance of the algorithm in the case of elephant flows by bulk traffic. We directly download a 100 MB file from the server to test the performance of MPTCP, MPTCP-BBR, and BCCPS. We compare the normalized download time and average MPTCP OFO queue size of MPTCP, MPTCP-BBR, and BCCPS. The results are illustrated in Fig. 16.

Fig. 16 shows that, in heterogeneous scenario, the BBR-based congestion control algorithms, MPTCP-BBR, achieves greater goodput and less download time than legacy MPTCP. Besides, compared to MPTCP-BBR, BCCPS further shortens download time because of our new scheduling algorithm. Although the scheduling algorithms of MPTCP and MPTCP-BBR are both LRF, the average MPTCP OFO queue size of MPTCP-BBR is still larger than that of MPTCP, due to the goodput gain brought by BBR. Compared with MPTCP and MPTCP-BBR, BCCPS reduces the average MPTCP OFO queue size.

3) Web Traffic: We further test the performance of the algorithm in the case of mouse flows by web traffic. We choose 3 different web traffic: Instagram, Amazon, and Wikipedia. The average file size is sorted as Instagram >Amazon > Wikipedia. Then compare the normalized download time and average MPTCP OFO queue size between MPTCP, MPTCP-BBR, and BCCPS. As shown in Fig. 17, BCCPS performs the best because it reduces overall download time and average MPTCP OFO queue through redundant transmission and fine-grained packet scheduling. Besides, as the size of the web traffic increases, we can see that BCCPS's advantages in performance are more obvious.

V. CONCLUSION AND FUTURE WORK

In this paper, in order to improve the performance of MPTCP in heterogeneous wireless networks, by taking the advantages of BBR and for considering bottleneck fairness, we proposed a BBR-based Congestion Control and Packet Scheduling (BCCPS) for Multipath TCP which focuses on reducing the completion time of mouse flows and providing excellent goodput to elephant flows. BCCPS relies on two new mechanisms: a well-designed BBR-based congestion control to maximize the goodput over all available paths, and a fine-grained packet scheduling algorithm. Using these mechanisms, BCCPS monitors and analyzes the dynamic network environment in real

Authorized licensed use limited to: Kongu Engineering College. Downloaded on June 29,2024 at 06:23:07 UTC from IEEE Xplore. Restrictions apply.

time and estimates each transmission path's quality. Based on the output of the path quality evaluation, BCCPS intelligently detects the shared bottleneck and balances the congestion among the subflows while keeping fairness with the single flow shared the bottleneck. Besides, BCCPS employs a two phase packet scheduling scheme to maximize the goodput for elephant flows while reduce the end-to-end delay for mouse flows. The two phase packet scheduling scheme improves data delivery efficiency. The simulation results and real network tests demonstrate that the proposed BCCPS offers better performance than default MPTCP in terms of goodput and end-to-end delay.

In the future, we will consider more complex simulation scenarios to analyze the performance of the proposed algorithms and further evaluate our scheme in real scenarios. Meanwhile, we will further refine the scheme design to consider fine-grained scheduling policies for different types of transmitted data.

REFERENCES

[1] M. Allman, V. Paxson, and E. Blanton, "TCP congestion control," 2009, RFC 5681, Accessed: Dec. 3, 2020. [Online]. Available: https://www.ietf.org/rfc/rfc5681.txt

[2] W. Lee, J. Koo, Y. Park, and S. Choi, "Transfer time, energy, and quota-aware multi-RAT operation scheme in smartphone," *IEEE Trans. Veh. Technol.*, vol. 65, no. 1, pp. 307–317, Jan. 2016.

[3] X. Zheng, Z. Cai, J. Li, and H. Gao, "A study on application-aware scheduling in wireless networks," *IEEE Trans. Mobile Comput.*, vol. 16, no. 7, pp. 1787–1801, Jul. 2017.

[4] B. Jin, S. Kim, D. Yun, H. Lee, W. Kim and Y. Yi, "Aggregating LTE and Wi-Fi: Toward intra-cell fairness and high TCP performance," *IEEE Trans. Wireless Commun.*, vol. 16, no. 10, pp. 6295–6308, Oct. 2017.

[5] J. Wu, B. Cheng, M. Wang, and J. Chen, "Energy-aware concurrent multipath transfer for real-time video streaming over heterogeneous wireless networks," *IEEE Trans. Circuits Syst. Video Technol.*, vol. 28, no. 8, pp. 2007–2023, Aug. 2018.

[6] M. G. Kibria, K. Nguyen, G. P. Villardi, K. Ishizu and F. Kojima, "Next generation new radio small cell enhancement: Architectural options, functionality and performance aspects," *IEEE Wirel. Commun.*, vol. 25, no. 4, pp. 120–128, Aug. 2018.

[7] A. Ford, C. Raiciu, M. Handley, and O. Bonaventure, "TCP extensions for multipath operation with multiple addresses," 2013, RFC 6824, Accessed: Dec. 3, 2020. [Online]. Available: https://www.ietf.org/rfc/rfc6824.txt

[8] Y. Xing, J. Han, K. Xue, J. Liu, M. Pan, and P. Hong, "MPTCP meets big data: Customizing transmission strategy for various data flows," *IEEE Netw.*, vol. 34, no. 4, pp. 35–41, Jul./Aug. 2020.

[9] H. Sinky, B. Hamdaoui, and M. Guizani, "Seamless handoffs in wireless hetnets: Transport-layer challenges and multi-path TCP solutions with cross-layer awareness," *IEEE Netw.*, vol. 33, no. 2, pp. 195–201, Mar./Apr. 2019.

[10] J. Xu, B. Ai, L. Chen, L. Pei, Y. Li and Y. Y. Nazaruddin, "When high-speed railway networks meet multipath TCP: Supporting dependable communications," *IEEE Wireless Commun. Lett.*, vol. 9, no. 2, pp. 202–205, Feb. 2020.

[11] D. Wischik, C. Raiciu, A. Greenhalgh, and M. Handley, "Design, implementation and evaluation of congestion control for multipath TCP," in *Proc. 8th USENIX Symp. Netw. Syst. Des. Implementation*, 2011, pp. 99–122.

[12] C. Raiciu, M. Handley, and D. Wischik, "Coupled congestion control for multipath transport protocols," 2011, RFC 6356, Accessed: Dec. 3, 2020. [Online]. Available: https://www.ietf.org/rfc/rfc6356.txt

[13] 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.

[14] 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.

[15] S. Ferlin-Oliveira, T. Dreibholz, and Ö. Alay, "Tackling the challenge of bufferbloat in multi-path transport over heterogeneous wireless networks," in *Proc. 22nd IEEE Int. Symp. Qual. Serv.*, 2014, pp. 123–128.

[16] H. Im, C. Joo, T. Lee, and S. Bahk, "Receiver-side TCP countermeasure to bufferbloat in wireless access networks," *IEEE Trans. Mobile Comput.*, vol. 15, no. 8, pp. 2080–2093, Aug. 2016.

[17] 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, 2016.

[18] E. Atxutegi, F. Liberal, H. K. Haile, K. Grinnemo, A. Brunstrom, and A. Arvidsson, "On the use of TCP BBR in cellular networks," *IEEE Commun. Mag.*, vol. 56, no. 3, pp. 172–179, Mar. 2018.

[19] N. Kuhn, E. Lochin, A. Mifdaoui, G. Sarwar, O. Mehani, and R. Boreli, "DAPS: Intelligent delay-aware packet scheduling for multipath transport," in *Proc. IEEE Int. Conf. Commun.*, 2014, pp. 1222–1227.

[20] S. Ferlin, Ö. Alay, O. Mehani, and R. Boreli, "BLEST: Blocking estimation-based MPTCP scheduler for heterogeneous networks," in *Proc. IEEE IFIP Netw. Conf.*, 2016, pp. 431–439.

[21] Y. Cao, M. Xu, and X. Fu, "Delay-based congestion control for multipath TCP," in *Proc. 20th IEEE Int. Conf. Netw. Protoc.*, 2012, pp. 1–10.

[22] B. Mao *et al.*, "Routing or computing? The paradigm shift towards intelligent computer network packet transmission based on deep learning," *IEEE Trans. Comput.*, vol. 66, no. 11, pp. 1946–1960, Nov. 2017.

[23] B. Mao *et al.*, "A novel non-supervised deep-learning-based network traffic control method for software defined wireless networks," *IEEE Wirel. Commun.*, vol. 25, no. 4, pp. 74–81, Aug. 2018.

[24] J. Pei, P. Hong, K. Xue, D. Li, D. S. Wei, and F. Wu, "Two-phase virtual network function selection and chaining algorithm based on deep learning in SDN/NFV-enabled networks," *IEEE J. Sel. Areas Commun.*, vol. 38, no. 6, pp. 1102–1117, Jun. 2020.

[25] W. Li, H. Zhang, S. Gao, C. Xue, X. Wang, and S. Lu, "SmartCC: A reinforcement learning approach for multipath TCP congestion control in heterogeneous networks," *IEEE J. Sel. Areas Commun.*, vol. 37, no. 11, pp. 2621–2633, Nov. 2019.

[26] Z. Xu, J. Tang, C. Yin, Y. Wang, and G. Xue, "Experience-driven congestion control: When multi-path TCP meets deep reinforcement learning," *IEEE J. Sel. Areas Commun.*, vol. 37, no. 6, pp. 1325–1336, Jun. 2019.

[27] T. Zhang and S. Mao, "Machine learning for end-to-end congestion control," *IEEE Commun. Mag.*, vol. 58, no. 6, pp. 52–57, Jun. 2020.

[28] M. Polese, F. Chiariotti, E. Bonetto, F. Rigotto, A. Zanella, and M. Zorzi, "A survey on recent advances in transport layer protocols," *IEEE Commun. Surveys Tut.*, vol. 21, no. 4, pp. 3584–3608, Oct.–Dec. 2019.

[29] C. Paasch, S. Ferlin, O. Alay, and O. Bonaventure, "Experimental evaluation of multipath TCP schedulers," in *Proc. ACM SIGCOMM Workshop Capacity Sharing Workshop*, 2014, pp. 27–32.

[30] C. Xu, Z. Li, J. Li, H. Zhang, and G.-M. Muntean, "Cross-layer fairness-driven concurrent multipath video delivery over heterogeneous wireless networks," *IEEE Trans. Circuits Syst. Video Technol.*, vol. 25, no. 7, pp. 1175–1189, Jul. 2015.

[31] Y. E. Guo, A. Nikravesh, Z. M. Mao, F. Qian, and S. Sen, "Accelerating multipath transport through balanced subflow completion," in *Proc. ACM Int. Conf. Mobile Comput. Netw.*, 2017, pp. 141–153.

[32] H. Zhang, W. Li, S. Gao *et al.* "Reles: A neural adaptive multipath scheduler based on deep reinforcement learning," in *Proc. 38th IEEE Int. Conf. Comput. Commun.*, 2019, pp. 1648–1656.

[33] S. Hassayoun, J. Iyengar, and D. Ros, "Dynamic window coupling for multipath congestion control," in *Proc. 20th IEEE Int. Conf. Netw. Protoc.*, 2011, pp. 341–352.

[34] W. Wei, K. Xue, J. H. Han, D. S. 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.

[35] D. Scholz, B. Jaeger, L. Schwaighofer, D. Raumer, F. Geyer, and G. Carle, "Towards a deeper understanding of TCP BBR congestion control," in *Proc. IFIP Netw. Conf. Workshops.*, 2018, pp. 1–9.

[36] S. Ma, J. Jiang, W. Wang, and B. Li, "Fairness of congestion-based congestion control: Experimental evaluation and analysis," 2017, *arXiv:1706.09115*. [Online]. Available: https://arxiv.org/abs/1706.09115

[37] M. Hock, R. Bless, and M. Zitterbart, "Experimental evaluation of BBR congestion control," in *Proc. 25th IEEE Int. Conf. Netw. Protoc.*, 2017, pp. 1–10.

[38] Y. Tao, J. Jiang, S. Ma, L. Wang, W. Wang, and B. Li, "Unraveling the RTT-fairness problem for BBR: A queueing model," in *Proc. IEEE Glob. Commun. Conf*, 2018, pp. 1–6.

[39] S. Abbasloo, Y. Xu, and H. J. Chao, "C2TCP: A flexible cellular TCP to meet stringent delay requirements," *IEEE J. Sel. Areas Commun.*, vol. 37, no. 4, pp. 918–932, Apr. 2019.

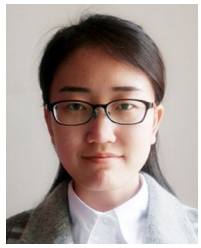
[40] M. Kheirkhah, I. Wakeman, and G. Parisis, "MMPTCP: A multipath transport protocol for data centers," in *Proc. 35th IEEE Int. Conf. Comput. Commun.*, 2016, pp. 1–9.

Authorized licensed use limited to: Kongu Engineering College. Downloaded on June 29,2024 at 06:23:07 UTC from IEEE Xplore. Restrictions apply.

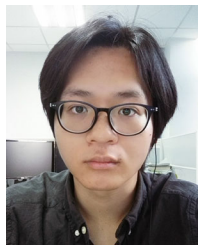
[41] C.-Y. Hong, M. Caesar, and P. B. Godfrey, "Finishing flows quickly with preemptive scheduling," *ACM SIGCOMM Comput. Commun. Rev.*, vol. 42, no. 4, pp. 127–138, 2012.
[42] A. Munir *et al.*, "Minimizing flow completion times in data centers," in *Proc. 32th IEEE Int. Conf. Comput. Commun.*, 2013, pp. 2157–2165.
[43] A. Munir *et al.*, "PASE: Synthesizing existing transport strategies for near-optimal data center transport," *IEEE/ACM Trans. Netw.*, vol. 25, no. 1, pp. 320–334, Feb. 2016.
[44] "NS3 simulator," Accessed: Dec. 3, 2020. [Online]. Available: https://www.nsnam.org/
[45] "MPTCP NS3 code," Accessed: Dec. 3, 2020. [Online]. Available: http://code.google.com/p/mptcp-ns3/
[46] Q. De Coninck, M. Baerts, B. Hesmans, and O. Bonaventure, "Observing real smartphone applications over multipath TCP," *IEEE Commun. Mag.*, vol. 54, no. 3, pp. 88–93, Mar. 2016.

Wenjia Wei (Member, IEEE) received the bachelor's degree from the School of Information Science and Engineering in 2013. He is currently working toward the Ph.D. degree in information and communication engineering with the Department of Electronic Engineering and Information Science (EEIS), University of Science and Technology of China . His research interests include future Internet architecture design and transmission optimization.

Kaiping Xue (Senior Member, IEEE) received the bachelor's degree from the Department of Information Security, University of Science and Technology of China (USTC), in 2003 and the Ph.D. degree from the Department of Electronic Engineering and Information Science (EEIS), USTC, in 2007. From May 2012 to May 2013, he was a Postdoctoral Researcher with Department of Electrical and Computer Engineering, University of Florida. He is currently a Professor with the School of Cyber Security and the Department of EEIS, USTC. His research interests include next-generation Internet, distributed networks and network security. He has authored and coauthored more than 80 technical papers in the areas of communication networks and network security. His work won Best Paper Awards in IEEE MSN 2017, IEEE HotICN 2019, and Best Paper Runner-up Award in IEEE MASS 2018. He is on the Editorial Board of several journals, including the IEEE TRANSACTIONS ON WIRELESS COMMUNICATIONS (TWC), the IEEE TRANSACTIONS ON NETWORK AND SERVICE MANAGEMENT (TNSM), and *Ad Hoc Networks*. He is an IET Fellow.



Jiangping Han received the bachelor's degree from the Department of Electronic Engineering and Information Science (EEIS), University of Science and Technology of China, in 2016. She is currently working toward the Ph.D. degree in communication and information systems with the Department of Electronic Engineering and Information Science, University of Science and Technology of China. Her research interests include future Internet architecture design and transmission optimization.



Yitao Xing (Graduate Student Member, IEEE) received the bachelor's degree in information security from the School of the Gifted Young, University of Science and Technology of China (USTC), in 2018. He is currently a Graduated Student in communication and information system with the Department of Electronic Engineering and Information Science (EEIS), USTC. His research interests include future Internet architecture and transmission optimization.

David S. L. Wei (Senior Member, IEEE) received the Ph.D. degree in computer and information science from the University of Pennsylvania in 1991. From May 1993 to August 1997, he was on the Faculty of Computer Science and Engineering with the University of Aizu, Japan (as an Associate Professor and then a Professor). He has authored and coauthored more than 100 technical papers in various archival journals and conference proceedings. He is currently a Professor of Computer and Information Science Department with Fordham University. His research interests include cloud computing, big data, IoT, and cognitive radio networks. He was a Guest Editor or a Lead Guest Editor for several Special Issues in the IEEE JOURNAL ON SELECTED AREAS IN COMMUNICATIONS, the IEEE TRANSACTIONS ON CLOUD COMPUTING and the IEEETRANSACTIONS ON BIG DATA. He also was an Associate Editor for IEEE TRANSACTIONS ON CLOUD COMPUTING, 2014–2018, and an Associate Editor for *Journal of Circuits, Systems and Computers*, 2013–2018.

Peilin Hong received the B.S. and M.S. degrees from the Department of Electronic Engineering and Information Science (EEIS), University of Science and Technology of China (USTC), in 1983 and 1986, respectively. She is currently a Professor and Advisor for Ph.D. candidate with the Department of EEIS, USTC. Her research interests include next-generation Internet, policy control, IP QoS, and information security. She has authored or coauthored two books and more than 150 academic papers in several journals and conference proceedings.

Authorized licensed use limited to: Kongu Engineering College. Downloaded on June 29,2024 at 06:23:07 UTC from IEEE Xplore. Restrictions apply.