

# Leveraging Coupled BBR and Adaptive Packet Scheduling to Boost MPTCP

Jiangping Han, Kaiping Xue, *Senior Member, IEEE*, Yitao Xing, Jian Li, Wenjia Wei, David S.L. Wei, *Senior Member, IEEE*, Guoliang Xue, *Fellow, IEEE*

**Abstract**—Multipath TCP (MPTCP) utilizes multiple paths for simultaneous data transmission to enhance performance. However, existing MPTCP protocols are still far from satisfactory in wireless networks because of their loss-based congestion control and the difficulty of managing multiple subflows. To overcome these problems, we redesign the coupled congestion control algorithm and scheduler to boost MPTCP in wireless heterogeneous networks. The main purpose is to promote transmission rate under lossy networks, while also provide stability when networks suffer physical link changes and asymmetric links. In this paper, inspired by Bottleneck Bandwidth and Round-trip propagation time (BBR), we first propose Coupled BBR that utilizes detected bandwidth to adjust the sending rate within an MPTCP connection. Coupled BBR provides high loss tolerance as well as balanced congestion among MPTCP subflows. Then, to further improve the performance, we propose an Adaptively Redundant and Predictive packet (AR&P) scheduler to improve adaptability and keep in-order packet delivery in highly dynamic network scenarios. Based on Linux kernel implementation and experiments in both testbed and real network scenarios, we show that the proposed scheme not only provides high throughput in wireless networks, but also improves robustness and reduces out-of-order packets in some harsh circumstances.

**Index Terms**—MPTCP, Congestion control, Scheduler, Wireless networks.

## I. INTRODUCTION

Multipath TCP (MPTCP) [1] is an emerging transport protocol, which enables the full use of the device's multiple interfaces and transmits data via multiple paths concurrently [2], [3]. MPTCP establishes subflows on available paths such that each subflow acts as a separate TCP flow. Based on TCP, MPTCP aims at providing higher transmission efficiency, stronger robustness, and better mobility support [4]. Till now, MPTCP has got some deployment [5] in real networks, and there have been some devices and applications, such as Apple Siri [6], in support of MPTCP.

Several schemes such as coupled congestion control algorithms [7]–[11] and scheduling algorithms [12]–[15] have

J. Han, K. Xue and W. Wei are with the Department of Electronic Engineering and Information Science, University of Science and Technology of China, Hefei 230027, China.

Y. Xing and J. Li are with the School of Cyber Security, University of Science and Technology of China, Hefei 230027, China.

K. Xue is also with the School of Cyber Security, University of Science and Technology of China, Hefei 230027, China.

D. Wei is with the Computer and Information Science Department, Fordham University, Bronx 10458, USA.

G. Xue is with the School of Computing, Informatics, and Decision Systems Engineering, Arizona State University, Tempe, AZ 85287, USA.

Corresponding Author: K. Xue (e-mail: kpxue@ustc.edu.cn).

been proposed to make MPTCP more practical. However, MPTCP is still not able to achieve the desired performance in wireless networks, which have a large number of random packet loss and rapidly changing link conditions. On the one hand, traditional loss-based congestion control algorithms can hardly make the best use of the full available bandwidth in lossy networks [16]. On the other hand, a fixed scheduler can not meet the ever-changing network conditions, where the unpredictable degradation in a single subflow may severely degrade the performance of other subflows in an MPTCP connection [17], [18]. Based on these facts, it is hard to achieve satisfactory end-to-end transmission performance for MPTCP [19].

Coupled congestion control algorithms (e.g., LIA, OLIA, BALIA) in MPTCP have been designed based on traditional TCP congestion control algorithm (for example, NewReno [20]) and treat packet loss as an indicator of congestion and decrease their congestion window when packet loss occurs. In today's network environment where wireless links are used frequently and random packet loss caused by physical links is common, it is hard for MPTCP to achieve the desired performance. MPTCP needs to change its way for transmission control for better performance. Among some state-of-art congestion control algorithms [21], [22], BBR shows its potential in lossy scenarios, which can make the best use of available bandwidth even when there is random packet loss. Inspired by BBR, we design a novel coupled congestion control algorithm and propose a customized scheduler for it. In this work, we mainly focus on two issues: 1) **Promote MPTCP in wireless lossy networks**, as well as provide high loss tolerance and achieve fairness and balanced congestion, and 2) **Further improve the transmission efficiency and stability of MPTCP in ever-changing and asymmetric networks** by designing a customized scheduler that suitable for the novel congestion control algorithm that precisely controls the multipath transmission.

We first design a novel coupled congestion control algorithm for MPTCP, called Coupled BBR, which is based on TCP BBR but is modified for MPTCP to achieve better performance. Coupled BBR follows the same mechanism of periodic bandwidth detection in convention BBR to provide high bandwidth utilization. In order to achieve the goals of fairness and balanced congestion for MPTCP defined in RFCs [1], [8], Coupled BBR sets the sending rate of each subflow differently. RFC 6356 [8] points out that running an uncoupled congestion control algorithm on each subflow makes an MPTCP flow unfairly take up more capacity compared with a single path

TCP flow, which means aggregated bandwidth of MPTCP should be no more aggressive than that of a single path TCP flow on the best available path. To achieve this goal, which is different from previous algorithms [7], [23], [24] that modify the increase function of the Additive Increase Multiplicative Decrease (AIMD) [25] scheme, Coupled BBR utilizes its measured bandwidth of all subflows to control each subflow's sending rate and achieve fairness to single-path TCP BBR flows. Besides, Coupled BBR also utilizes a data allocation rate based on the bandwidth measurement results, therefore it can better balance congestion among subflows.

Secondly, based on the real-time measurement and steady sending rate of Coupled BBR, we propose an Adaptively Redundant and Predictive packet (AR&P) scheduler to enhance MPTCP performance in highly dynamic and asymmetric networks. Two scheduling methods are included in AR&P scheduler, 1) Adaptively Redundant Scheduling (AR-Scheduling), and 2) Predictive packet Scheduling (P-Scheduling). AR-Scheduling is designed to achieve high goodput and low latency in different network scenarios, and provides better adaptability in highly dynamic scenarios. It adaptively decides whether to send redundant packets on each subflow according to the real-time path conditions. By sending redundant packets on subflows with low bandwidth and high RTT, AR-Scheduling is able to provide better flexibility when the network environment changes rapidly. Besides, P-Scheduling is designed to reduce out-of-order packets in asymmetric networks. Different from previous packet schedulers which only schedule in each congestion window, P-Scheduling calculates the arrival time of packets and schedules each packet one-by-one. Taking the advantages of Coupled BBR's steady sending rate and smooth transmission, P-Scheduling could accurately control the arrival time of each packet, thereby reducing out-of-order packets significantly.

To summarize, in this paper we present Coupled BBR and AR&P Scheduler for MPTCP. With our scheme, the performance of MPTCP is enhanced in lossy, dynamic, and asymmetric networks. The main contributions of this paper are as follows:

- We propose Coupled BBR as a coupled congestion control algorithm for MPTCP to obtain better performance in wireless lossy networks. Coupled BBR provides high bandwidth utilization and stable sending rate, while also achieving fairness to TCP BBR flows and balancing congestion among MPTCP subflows.
- Based on Coupled BBR, AR&P Scheduler is proposed to further help MPTCP for managing multipath transmission. It includes two scheduling methods: 1) AR-Scheduling automatically chooses whether to send redundant packets according to real-time path conditions, in order to provide better adaptability in highly dynamic networks. 2) P-Scheduling schedules each packet according to its arrival time, which keeps packets arriving in order and reduces out-of-order packets in asymmetric networks.
- Coupled BBR and AR&P Scheduler are implemented in MPTCP Linux kernel v0.94 [26] and tested in both testbed and real networks. Extensive results show that

our scheme gives MPTCP a higher elasticity, making it more feasible in today's networks.

The rest of this paper is organized as follows: Section II introduces the background and motivation of our work. We present our design and the details of each algorithm in Section III, Section IV, and Section V. The implementation and evaluation are shown in Section VI. Sections VII and VIII show the related work and the discussion of our work, respectively. Finally, Section IX draws the conclusion.

## II. BACKGROUND AND MOTIVATION

We first take a brief overview of MPTCP and BBR. Then, we discuss the opportunities and challenges that BBR brings to MPTCP.

### A. Overview of MPTCP

MPTCP is a multipath transport protocol proposed by IETF [1]. As an extension of TCP, it provides reliable transmission service, while also enables multipath transmission to gain better performance. MPTCP inherits the drawbacks of conventional TCP, which are mainly caused by traditional loss-based congestion control algorithms. They treat packet loss as a signal of congestion and halve the congestion window when packet loss occurs, which leads to poor performance and causes fluctuation of sending rate in lossy scenarios such as wireless networks [27]–[29].

Besides, there are some new issues introduced by multipath transmission in MPTCP. MPTCP needs to be friendly to TCP flows, which means an MPTCP flow should not be more aggressive than a single-path TCP flow on the best path [8]. Also, MPTCP needs to balance congestion, which means to migrate data from congested subflows to less congested ones [30]. Moreover, MPTCP should achieve stronger robustness. When some subflows fail, MPTCP is supported to keep running since it can transfer data on other available subflows. Meanwhile, MPTCP also needs to reduce out-of-order packets, which is caused by different RTTs among subflows in asymmetric networks.

### B. Overview of BBR

Different from traditional loss-based congestion control, BBR measures the bandwidth and RTT of the bottleneck which a flow goes through [31]. Then based on the measurement, it adjusts the sending rate to make the best use of the bottleneck bandwidth. BBR keeps high throughput in lossy networks and maintains a smooth rate during the transmission. Through other popular congestion control algorithms like Cubic make a faster recovery for high throughput in lossy scenarios, they create fluctuating sending rate, and provide much worse performance than BBR when suffering high loss rate [22]. Additionally, the use of BBR stops creating queues in the network, thereby reducing RTT and leading to low transmission delay.

Specifically, BBR periodically measures bottleneck bandwidth and adjusts the transmission rate at its PROBE\_BW phase, which accounts for the vast majority (i.e., almost

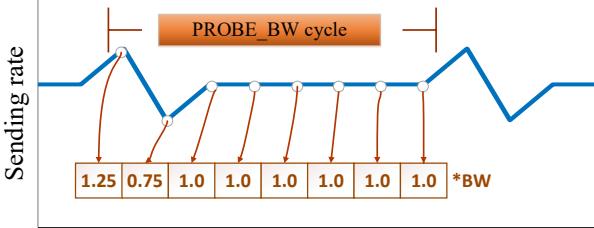


Fig. 1. Sending rate of BBR in RROBE\_BW phase.

98%) of its running time [31]. As shown in Fig. 1, BBR treats 8 RTTs as a cycle during the PROBE\_BW phase. In each RTT of a cycle, BBR sends data as a rate of  $pacing\_rate = pacing\_gain \cdot BW$ , where  $pacing\_gain = (1.25, 0.75, 1.00, 1.00, 1.00, 1.00, 1.00, 1.00)$  in each RTT respectively. In this state,  $BW$  is the maximum measured value of delivery rate during a period of time, which is noted as an estimated result of bottleneck bandwidth. During the first RTT, BBR increases the sending rate to  $1.25 \cdot BW$  to probe the remaining available bandwidth, and during the second RTT, it reduces the rate to  $0.75 \cdot BW$  to drain the queues that may be created in the previous RTT. After the first two RTTs, BBR keeps sending data smoothly using the detected bandwidth for 6 RTTs. In this process, the congestion window ( $cwnd$ ) is no longer the deciding factor, it is  $pacing\_rate$  instead. BBR sets the interval time between two packets to  $packet\_size/pacing\_rate$  so as to control the sending rate and keep the transmission smooth. For each 10 s, BBR goes through a PROBE\_RTT phase, keeps inflight to 4 for  $\max(RTT, 0.2sec)$  to probe minimum RTT of the path.

### C. MPTCP over BBR: Opportunities and Challenges

Considering the superiority of BBR, MPTCP can be promoted simply by replacing its congestion control algorithm with BBR. We measure the performance of MPTCP with conventional BBR in a lab-built platform as shown in Fig. 2. Our testbed includes a pair of MPTCP server and client, two pairs of TCP servers and clients, and four routers within the topology shown in Fig. 2. MPTCP connection includes two subflows, where each subflow passes through two routers. The links between two routers represent the bottleneck in the network. The links between client and router or between server and router do not affect the transmission. Both the bottleneck links have 100 Mbps bandwidth and 25 ms delay. At each bottleneck, there are two TCP background flows on each path using the same kind of congestion control algorithm as MPTCP uses.

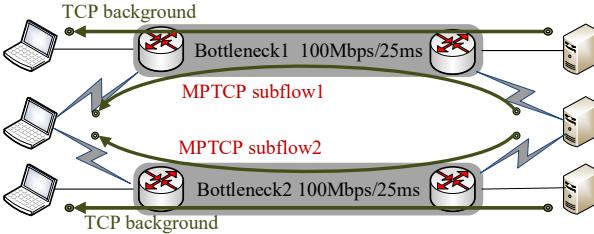


Fig. 2. Topology of the testbed.

Table I shows the throughput performance in lossy scenarios, where the random packet loss rate of each subflow is 0.01% (subflow<sub>1</sub>) and 0.1% (subflow<sub>2</sub>), respectively. We observe that although packet loss rates of 0.01% and 0.1% are not too high in the actual wireless networks, the throughput of original algorithms still drops dramatically. Among them, Cubic is better than others but is unable to sustain its superiority when the packet loss rate goes up. Moreover, the bandwidth utilization of LIA, OLIA, BALIA, and Cubic is much lower than the available bandwidth that an ideal congestion control algorithm could achieve.

TABLE I  
AVERAGE THROUGHPUT

Throughput	BBR	Cubic	LIA	OLIA	BALIA
MPTCP (Mbps)	55.9	35.1	20.3	19.9	20.9
Subflow 1 (Mbps)	30.7	27.6	19.1	19.6	20.5
Subflow 2 (Mbps)	25.2	7.5	1.2	0.3	0.6
TCP on path 1 (Mbps)	31.9	27.7	21.1	21.3	20.8
TCP on path 2 (Mbps)	27.5	8.5	1.1	1.0	1.2
Bandwidth utilization	88%	54%	32%	32%	33 %

Fig. 3 shows the RTT distribution, where the random loss rates of subflow<sub>1</sub> and subflow<sub>2</sub> is 0 and 0.01% respectively. BBR keeps RTT of MPTCP concentrating at around 55 ms. But half of RTTs of other algorithms are concentrated at the zone of 85 ms, which corresponds to the subflow with no random packet loss. Since traditional congestion control algorithms increase the congestion window and fill the buffer of the intermediate routers until packet loss, packets are queued at the routers for a long time, resulting in longer RTT. On the contrary, BBR does not cause network overload and keeps RTT low.

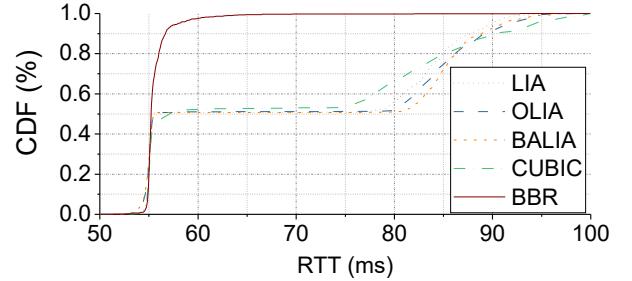


Fig. 3. RTT distribution of MPTCP using different congestion control algorithms.

Compare with other congestion algorithms, BBR makes each subflow obtain high throughput in lossy networks and keeps low RTT in congested networks. However, the original BBR treats multiple subflows of an MPTCP connection as separate flows that work independently rather than a unified connection. Thus the goals of fairness and balanced congestion can not be achieved. To achieve these goals, we provide a new algorithm that utilizes bandwidth detection for coupled congestion control, which is called Coupled BBR. In addition, a functional scheduler also needs to be further designed. Previous schedulers are usually based on the congestion window, while Coupled BBR changes it to smooth sending rate and makes previous schedulers no longer suitable. Thus, we propose a novel AR&P scheduler to promote the performance

in ever-changing and asymmetric networks on the basis of Coupled BBR.

### III. SYSTEM DESIGN

In this section, we introduce our design of congestion control and scheduler for MPTCP. Fig. 4 illuminates the framework. The framework basically includes two parts: a coupled congestion control algorithm called Coupled BBR and a novel scheduler called Adaptively Redundant and Predictive packet (AR&P) scheduler.

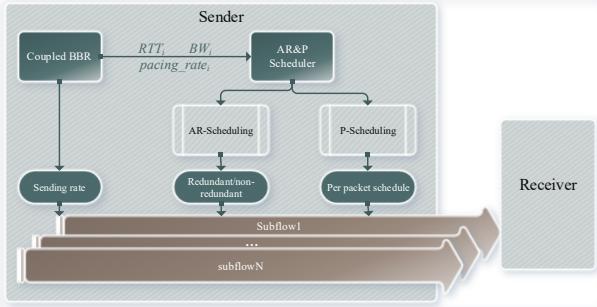


Fig. 4. Coupled BBR and AR&P Scheduler for MPTCP

Coupled BBR and AR&P scheduler take on the functions of rate control and data scheduling, respectively. Coupled BBR performs the function of coupled congestion control for MPTCP, using measured bandwidth and RTT to control the sending rate of each subflow. It provides a steady and proper sending rate for MPTCP, ensures high throughput, and at the same time, achieves fairness to TCP BBR flows and balances congestion among MPTCP subflows. Coupled BBR shares its measured result with AR&P scheduler for further scheduling function. AR&P scheduler helps manage subflows by scheduling packets properly through subflows under various network conditions with the following two scheduling methods:

1) AR-Scheduling decides the redundant/non-redundant state of each subflow. If a subflow is in poor network conditions (low bandwidth or large RTT), AR-Scheduling tends to send redundant packets via it. Otherwise, the subflow is used to transmit non-redundant packets to aggregate bandwidth resources. By adjustment based on real-time measurement, AR-Scheduling improves robustness and guarantees high throughput in highly dynamic networks.

2) P-Scheduling schedules each packet to a target subflow according to the packet's arrival time. Each packet is scheduled to a subflow with the earliest arrival time to reduce out-of-order packets and improve performance in asymmetric scenarios.

Coupled BBR and AR&P scheduler are implemented at MPTCP sender for better transmission control. MPTCP receiver performs the original operation and does not need any other extra interaction with the sender. In the next sections, we present the details of each algorithm.

#### IV. COUPLED BBR

Coupled BBR retains most of the operations in the conventional BBR, periodically measures the bottleneck bandwidth

and uses the measured bandwidth to allocate the sending rate of each subflow. Instead of the previous way which adjusts the AIMD parameters, Coupled BBR modifies the PROBE\_BW phase and sets the sending rate directly according to the measurement results of each subflow to keep fairness and balanced congestion. And the same as the conventional BBR, Coupled BBR spends the vast majority of its time in PROBE\_BW phase (about 98 percent) [31], that makes it effective to achieve fairness and balanced congestion in the entire transmission process. In this way, Coupled BBR could keep a steady sending rate in PROBE\_BW phase and provides smooth transmission performance.

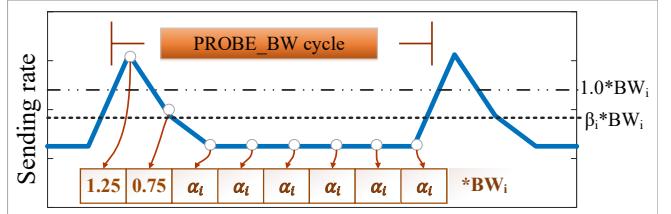


Fig. 5. Sending rate of Coupled BBR.

As shown in Fig. 5, Coupled BBR modifies *pacing\_gain*<sub>*i*</sub> for a subflow<sub>*i*</sub> to a cycle of {1.25, 0.75,  $\alpha_i$ ,  $\alpha_i$ ,  $\alpha_i$ ,  $\alpha_i$ ,  $\alpha_i$ ,  $\alpha_i$ ,  $\alpha_i$ } in the PROBE\_BW phase. The sending rate is *pacing\_gain*<sub>*i*</sub>.  $BW_i = \{1.25, 0.75, \alpha_i, \alpha_i, \alpha_i, \alpha_i, \alpha_i, \alpha_i, \alpha_i\} \cdot BW_i$ , where  $BW_i$  is the maximum detected available bandwidth of subflow<sub>*i*</sub>. The *pacing\_gain*<sub>*i*</sub> of {1.25, 0.75} for the first two RTTs is used to ensure the ability for each subflow to measure available bandwidth  $BW_i$ , that is the same as the conventional BBR. After that, for the next 6 RTTs, Coupled BBR replaces the *pacing\_gain* with a smaller parameter  $\alpha_i$  to achieve fairness and balanced congestion, which is related to the bandwidth of each subflow:

$$\alpha_i = \frac{4\beta_i - 1}{3}, \quad (1)$$

$$\beta_i = \frac{BW_i \cdot \max\{BW_i\}}{\sum_{j \in \mathcal{S}} BW_j^2}, \quad (2)$$

where  $S$  denotes the set of all subflows.

Coupled BBR is implemented at the sender side, and does not require interaction between the receiver and the sender. Algorithm 1 shows the algorithm of Coupled BBR. Consider that  $\alpha_i$  may be less than 0 because of Eq.(1), Coupled BBR sets the sending rate to  $4 \cdot \text{packets}/\text{RTT}_i$  if  $\alpha_i \leq 0$ , which is similar to the PROBE\_RTT phase. If a subflow enters this state, the 4 packets in an RTT round can be utilized to probe minimum RTT and protect the activity of a subflow. Next, we will show how to use  $\alpha_i$  to achieve the goals of MPTCP.

Assume that there are some TCP flows runs on subflow $_i$ 's path, and throughput of a single path TCP BBR flow is  $T_i^{TCP}$ . Let  $T_i^{MP}$  denotes the average throughput of MPTCP subflow $_i$ , and  $T^{MP}$  denotes the average throughput of a MPTCP connection, where  $T^{MP} = \sum_{i \in S} T_i^{MP}$ . Coupled BBR keeps the *pacing\_rate* of 1.25, 0.75 in the first two round and changes the *pacing\_rate* of the last 6 RTTs to  $\alpha_i$ . Therefore the average sending rate of subflow $_i$  is  $(1.25 + 0.75 + 6\alpha_i)BW_i/8 = \beta_i BW_i$ . Therefore the average throughput of MPTCP subflow $_i$  using Coupled BBR is:  $T_i^{MP} = \beta_i \cdot BW_i$ .

**Algorithm 1:** Coupled BBR

---

```

for each subflowi do
     $\beta_i = \frac{BW_i \cdot \max_j \{BW_j\}}{\sum_{j \in \mathcal{S}} BW_j^2}$ ;
     $\alpha_i = (4\beta_i - 1)/3$ ;
    pacing_gain = [1.25, 0.75,  $\alpha_i, \alpha_i, \alpha_i, \alpha_i, \alpha_i, \alpha_i$ ];
    if now >= nextSendTime then
        sendpacket();
        /* cycle_index is the index of RTT round in a
        PROBE_BW cycle */;
        if cycle_index > 2 then
            if  $\alpha_i > 0$  then
                nextSendTime = now + packet.size /
                    ( $\alpha_i \cdot BW_i$ );
            else
                nextSendTime = now + RTTi/4;
        else
            nextSendTime = now + packet.size /
                (pacing_gain[cycle_index] · BWi);

```

---

The average throughput of the overall MPTCP connection using Coupled BBR is:

$$\begin{aligned}
 T^{MP} &= \sum_{i \in \mathcal{S}} T_i^{MP} = \sum_{i \in \mathcal{S}} \beta_i \cdot BW_i \\
 &= \sum_{i \in \mathcal{S}} \frac{BW_i^2}{\sum_{j \in \mathcal{S}} BW_j^2} \cdot \max_{j \in \mathcal{S}} \{BW_j\} \\
 &= \max_{j \in \mathcal{S}} \{BW_j\}.
 \end{aligned}$$

This simply achieves the fairness between MPTCP and TCP BBR flows. Coupled BBR allocates a percentage of bandwidth to subflow<sub>i</sub> by the weight  $\beta_i$ . The overall MPTCP throughput is  $T^{MP} = \max_{i \in \mathcal{S}} \{BW_i\} = \max_{i \in \mathcal{S}} T_i^{TCP}$ , which equals the throughput of TCP BBR flows on the best path.

Moreover, Coupled BBR also has the ability to migrate data from congested paths and increase the data traffic on subflows with good path conditions. Given the information of bandwidth, Coupled BBR takes  $BW_i$  as a representation of the quality of a subflow<sub>i</sub>, and makes  $\beta_i$  of each subflow meets the following:  $\frac{\beta_i}{BW_i} = \frac{\beta_j}{BW_j}, \forall i, j$ . The data allocation is related to each subflow's detected bandwidth, and subflows with higher bandwidth carry more data traffic.

## V. ADAPTIVELY REDUNDANT AND PREDICTIVE PACKET SCHEDULER

Based on Coupled BBR, AR&P Scheduler is further proposed to promote MPTCP in asymmetric networks, as well as in the scenarios where path conditions are changing. AR&P scheduler includes two methods: 1) AR-Scheduling and 2) P-Scheduling. As shown in Fig. 6, AR-Scheduling first decides the redundant or non-redundant state of each subflow. Then, redundant packets are scheduled on subflows at redundant state. P-Scheduling works on the subflow in the non-redundant state, and schedules each packet according to the predictive arrival time.

### A. AR-Scheduling

By taking into account the real-time path condition measured by Coupled BBR, AR-Scheduling adaptively sends redundant packets on subflows with bad path conditions (e.g., low bandwidth and high RTT) to provide both high transmission robustness and flexibility in dynamic networks. When the user moves and the path conditions deteriorate, AR-Scheduling stops to send new packets on the subflow with bad path conditions and utilizes redundant packets to improve the reliability. The reason that utilizing redundant packets instead of stopping sending packets on the subflow with bad path conditions is to keep the activity of subflows. The redundant packets can be used to improve the robustness and detect the path conditions. Once the path conditions change for the better, AR-Scheduling returns to send non-redundant packets to improve bandwidth utilization efficiency.

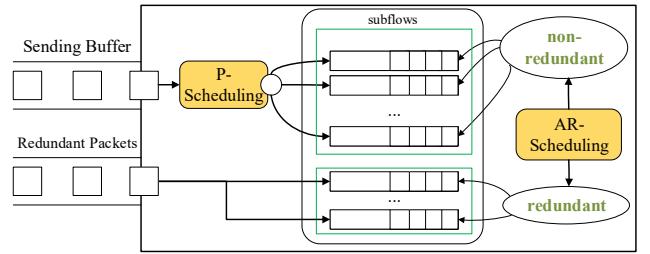


Fig. 6. AR&P Scheduler.

The path conditions of each subflow are measured in real-time by Coupled BBR. We use  $x_i$  and  $r_i$  to denote the sending rate and RTT of subflow<sub>i</sub>, respectively. Let  $\mathcal{N}$  denote the set of subflows in non-redundant state, and  $\mathcal{R} = \mathcal{S} - \mathcal{N}$  denote the set of subflows in redundant state. The goal of AR-Scheduling is choosing the non-redundant subflows  $\mathcal{N}$  to maximize goodput ( $J_1 = \log \sum_{i \in \mathcal{N}} x_i$ ) and minimize average RTT ( $J_2 = \log \sum_{i \in \mathcal{N}} x_i r_i / \sum_{i \in \mathcal{N}} x_i$ ). The objective of AR-Scheduling can be expressed as a multiple objectives utility function:

$$\max_{|\mathcal{N}| \geq 1} \log \sum_{i \in \mathcal{N}} x_i - \log \frac{\sum_{i \in \mathcal{N}} x_i r_i}{\sum_{i \in \mathcal{N}} x_i}, \quad (3)$$

where  $|\mathcal{N}| \geq 1$  means there should be at least one subflow to send non-redundant packets.

In order to reduce computational complexity, AR-Scheduling utilizes a greedy method:

1) Sort subflows in  $\mathcal{S}$  as  $i_1, \dots, i_n$ , where  $\frac{x_{i_1}}{r_{i_1}} \geq \dots \geq \frac{x_{i_n}}{r_{i_n}}$ . Add subflow<sub>i<sub>1</sub></sub> to  $\mathcal{N}$ .

There should be at least one subflow in  $\mathcal{N}$ . If  $\mathcal{N}$  only includes one subflow<sub>i</sub>, then Eq. (3) becomes  $x_i/r_i$ . AR-Scheduling chooses the subflow with the greatest  $x_i/r_i$  as the initial subflow in  $\mathcal{N}$ , where  $x_i/r_i$  denotes the subflow value.

2) For each  $j$  in  $2, \dots, n$ , add it to  $\mathcal{N}$  if the objective after adding  $i_j$  to  $\mathcal{N}$  is larger than that of the original  $\mathcal{N}$ , which means:

$$\begin{aligned}
 &\log \left( \sum_{i \in \mathcal{N}} x_i + x_{i_j} \right) - \log \left( \frac{\sum_{i \in \mathcal{N}} x_i r_i + x_{i_j} r_{i_j}}{\sum_{i \in \mathcal{N}} x_i + x_{i_j}} \right) \\
 &> \log \sum_{i \in \mathcal{N}} x_i - \log \frac{\sum_{i \in \mathcal{N}} x_i r_i}{\sum_{i \in \mathcal{N}} x_i}.
 \end{aligned}$$

To simplify it, we have  $\frac{x_{ij}}{\sum_{i \in \mathcal{N}} x_i} > \frac{\sum_{i \in \mathcal{N}} x_i(r_{ij} - 2r_i)}{\sum_{i \in \mathcal{N}} x_i r_i}$ . Otherwise, add it to  $\mathcal{R}$ .

The greedy method gives an optimal result when there are two subflows in an MPTCP connection. Considering that mobile devices typically have two interfaces (4G/5G and WiFi) to establish two subflows in practice, the greedy method can give the optimal strategy in most cases. In addition, we compare the solutions when there are three subflows in the simulation, where the result given by the greedy method is not very different from the optimal solution.

### Algorithm 2: AR-Scheduling

```

Input: Subflows,  $\mathcal{S} = \{1, \dots, n\}$ .
Output: Non-redundant set  $\mathcal{N}$  and redundant set  $\mathcal{R}$ .
Sort subflows as  $i_1, \dots, i_n$ , where  $\frac{x_{i_1}}{r_{i_1}} \geq \dots \geq \frac{x_{i_n}}{r_{i_n}}$ ;
 $\mathcal{N} = \{i_1\}$ ,  $\mathcal{R} = \emptyset$ ;
for each  $j \in \{2, \dots, n\}$  do
  if  $\frac{x_{ij}}{\sum_{i \in \mathcal{N}} x_i} \leq \frac{\sum_{i \in \mathcal{N}} x_i(r_{ij} - 2r_i)}{\sum_{i \in \mathcal{N}} x_i r_i}$  or  $\text{inflight}_i < 4$ 
    then
       $\mathcal{R} = \mathcal{R} \cup \{i_j\}$ ;
    else
       $\mathcal{N} = \mathcal{N} \cup \{i_j\}$ ;
```

In addition, AR-Scheduling utilizes inflight packets for the auxiliary judgment. If a packet loss occurs, the sender requires 3 duplicated ACKs to start fast retransmission. If the inflight packets are less than 4, the subflow will not start fast transmission but just wait for time-out retransmission. This packet loss on the single subflow may even decrease the throughput of other subflows. So when the inflight packets of  $\text{subflow}_i$  are less than 4, AR-scheduling marks that  $\text{subflow}_i$  as in the redundant state. The algorithm of AR-Scheduling is shown in Algorithm 2. To reduce computing overhead, AR-Scheduling makes decisions every  $\min_{i \in \mathcal{S}} r_i$  during the transmission.

### B. P-Scheduling

Based on the smooth sending rate of each subflow provided by Coupled BBR, P-Scheduling method predicts the arrival time of each packet and precisely controls each packet for better performance. P-Scheduling is a pre-scheduling method that schedules packets before they are sent. It schedules packets sequentially onto the appropriate subflow which gives the earliest arrival time of a packet to keep in-order packets arrival as well as reduce latency. As shown in Fig. 7, P-Scheduling pre-schedules the packets in the scheduling window to subflows for sending in the future. P-Scheduling method works as follows: When scheduling packet  $j$ , it calculates the arrival time of scheduling the new packet on each subflow, then chooses a subflow with the smallest arrival time and schedules packet  $j$  on the subflow. In this way, P-Scheduling ensures that the packets scheduled after packet  $j$  will not arrive earlier than packet  $j$ , therefore it keeps in-order packets arrival and the number of out-of-order packets can be significantly reduced in asymmetric networks.

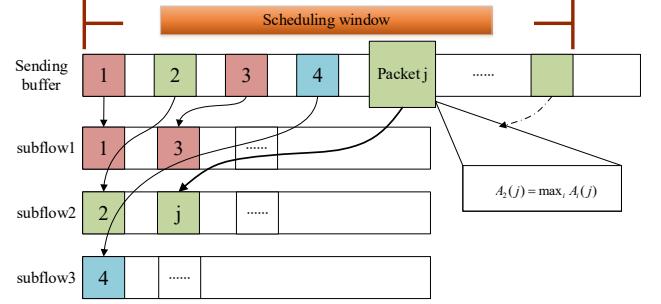


Fig. 7. Predictive packet scheduling

P-Scheduling pre-schedules packets on subflows. Therefore how many packets should be pre-scheduled is a key parameter. Scheduling all the packets in the sending buffer is not a good idea, since network conditions are changing and old prediction may not suitable for the new environments. P-Scheduling maintains a scheduling window at the size of  $\max_{i \in \mathcal{N}} RTT_i \cdot \sum_{i \in \mathcal{N}} BW_i$ . The scheduling window is set to ensure in-time scheduling as well as enough packets to schedule for each available subflow. Each packet in this window is scheduled to a certain subflow according to the predicted arrival time, and the packets outside the scheduling window will not be scheduled until they are included in the scheduling window.

Once packets in the scheduling window are sent out or the scheduling window size gets larger, new packets can be accommodated in the scheduling window and need to be scheduled by P-Scheduling. The set of new packets are noted as  $\{j_1, j_2, \dots, j_m\}$ . Then, P-Scheduling calculates the arrival time of each packet on available subflows, and schedules each packet to the subflow with the minimum arrival time in order.

Let  $A_i(j)$  denote the predicted arrival time of sending packet  $j$  on subflow  $i$ , and  $t_0$  denote the current moment. When MPTCP schedules the packet  $j$ , the set of packets that are already scheduled on subflow  $i$  but not sent out yet is  $\mathcal{L}_i$ . And the size of a packet  $j$  is  $s_j$ . P-Scheduling predicts the arrival time as:

$$A_i(j) = t_0 + \frac{\sum_{j' \in \mathcal{L}_i} s_{j'}}{x_i} + \frac{r_i}{2}, \quad (4)$$

where the second item  $\sum_{j \in \mathcal{L}_i} s_j / x_i$  on the right hand side of the equation is the waiting time for packet  $j_k$  to start transmitting if it is scheduled on subflow  $i$ . The third item  $r_i/2$  is the transmission time of each packet. The arrival time is the current moment  $t_0$  plus the waiting time and transmission time. When a packet is going to be scheduled, P-Scheduling chooses a subflow with the minimum  $A_i(j)$  to schedule packet  $j$  on it.

Algorithm 3 shows the algorithm of P-Scheduling. P-Scheduling only works in the PROBE\_BW state, which has a stable sending rate and takes up most of the time during the transmission. Otherwise, the sending rates of subflows are not stable, and P-Scheduling behaves the same as Round-Robin. When working with AR-Scheduling method, P-Scheduling only considers subflows in the non-redundant state and schedules new packets for them. If a subflow is in the

---

**Algorithm 3: P-Scheduling**


---

**Input:** New packets in the scheduling window:  
 $\mathcal{J} = \{j_1, \dots, j_m\}$ .  
**for** each packet  $j_k \in \mathcal{J}$  **do**  
  **for** each  $i \in \mathcal{N}$  **do**  
     $A_i(j_k) = t_0 + \frac{\sum_{j' \in \mathcal{L}_i} s_{j'}}{x_i} + \frac{r_i}{2}$ ;  
     $i_k = \arg \min_{i \in \mathcal{N}} A_i(j_k)$ ;  
    schedule packet  $j_k$  on the subflow  $i_k$ ;

---

redundant state, P-Scheduling will not schedule new packets on it.

**Error analysis:** Assume that the jitter of sending rate and RTT of subflow  $i$  is  $\Delta x_i$  ( $|\Delta x_i/x_i| < \epsilon_1$ ) and  $\Delta r_i$  ( $|\Delta r_i/r_i| < \epsilon_2$ ), respectively. The prediction error of  $A_i(j)$  is:

$$\Delta A_i(j) = \frac{\Delta x_i \sum_{j' \in \mathcal{L}_i} s_{j'}}{x_i(x_i + \Delta x_i)} + \frac{\Delta r_i}{2}.$$

Limited by the scheduling window,  $\sum_{j' \in \mathcal{L}_i} s_{j'} \in [0, x_i \cdot \max_{i \in \mathcal{N}} r_i]$ , and the average out-of-order packet  $O$  is:

$$\begin{aligned} O &= \max_{i \in \mathcal{N}} \sum_{i' \in \mathcal{N}/i} x_{i'} E\{\Delta A_i(j)\} \\ &\leq \frac{1}{2} \sum_{i \in \mathcal{N}} x_i \max_{i \in \mathcal{N}} r_i \left( \left| \frac{\max_{i \in \mathcal{N}} \Delta r_i}{\max_{i \in \mathcal{N}} r_i} \right| + \left| \frac{\Delta x_i}{x_i} \right| \right) \\ &\leq \frac{1}{2} \sum_{i \in \mathcal{N}} x_i \max_{i \in \mathcal{N}} r_i (\epsilon_1 + \epsilon_2). \end{aligned}$$

Therefore P-Scheduling can keep the out-of-ordered packets in a low level even there are jitters in the network environment.

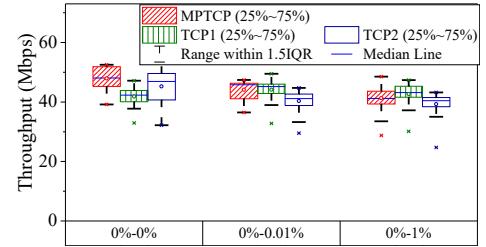
## VI. PERFORMANCE EVALUATION

We use both the simulation and real network measurement to evaluate the performance of Coupled BBR and AR&P scheduler. Traditional MPTCP congestion control algorithms (LIA, OLIA, BALIA) and schedulers (Round-Robin and min-RTT) are using as contrasts. To verify the performance of proposed schemes in different network scenarios, we integrate Coupled BBR and AR&P scheduler into MPTCP v0.94 implemented in Linux kernel and measure the performance in a lab-built testbed with 8 nodes and real network scenarios. In most of the tested scenarios, the performance of MPTCP can be improved by more than two-and-a-half times. In order to further evaluate the performance in larger network topologies, we also simulate the proposed algorithms under different network scenarios utilizing a packet-level simulator to show the simulation results.

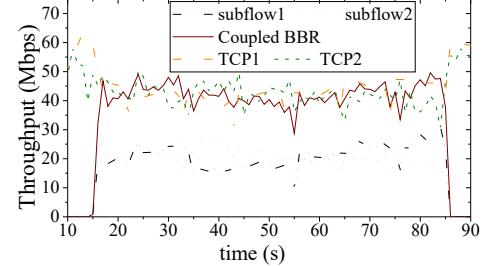
### A. Experiments in the Testbed

We integrate Coupled BBR and AR&P scheduler into MPTCP v0.94 implemented in Linux kernel [26] and test their performance in different scenarios. The testbed topology is the same as Fig. 2. Considering that the device usually has two interfaces (4G and Wi-Fi) in the real network, we also used two subflows in an MPTCP connection in the experiments.

Fig. 8 shows the performance of Coupled BBR in lossy networks. The bandwidth and delay of both bottlenecks are



(a) Different loss rate.



(b) loss rate of two paths: 0% - 1%.

Fig. 8. Coupled BBR in lossy networks.

100 Mbps and 25 ms, respectively. bottleneck<sub>2</sub> suffers varying random packet loss rates of 0%, 0.01%, and 1% in different scenarios, while bottleneck<sub>1</sub> does not have random packet loss. There are two TCP background flows at each bottleneck. Fig. 8(a) indicates that at different settings of path loss rate, Coupled BBR effectively achieves the goal of fairness, which gets the same throughput as that of a single-path TCP BBR flow on the best path. When the loss rate increases, the throughput of Coupled BBR decreases slightly but still achieves high throughput. Fig. 8(b) shows the real-time throughput when the loss rate of two subflows is 0% and 1%, respectively. Subflow<sub>2</sub> can still get a satisfactory throughput when the loss rate reaches 1%, and the sending rate keeps little fluctuation. In a word, MPTCP over Coupled BBR not only provides high throughput and less fluctuation in lossy networks but also achieves fairness to TCP BBR flows.

Fig. 9 shows the performance of Coupled BBR in asymmetric networks, where Fig. 9(a) and Fig. 9(b) show asymmetric bandwidth scenarios and asymmetric path delay scenarios, respectively. In Fig. 9(a), the loss rate and delay of both bottlenecks are 0% and 25 ms, the bandwidth of two path changes for different scenarios. Coupled BBR can achieve the same throughput as that of a single-path TCP BBR flow on the best path, and allocate more data on the best path to balance congestion. In Fig. 9(b), the bandwidth and loss rate of the two bottlenecks is set to 20 Mbps and 0%, the path delay changes for different scenarios. When the delay difference becomes larger, the throughput of MPTCP decreases slightly. Meanwhile, Coupled BBR allocates the same proportion of data to each subflow and still maintains fairness to TCP BBR flows. In summary, Coupled BBR achieves better loss tolerance and steady sending rate, while also achieves fairness to TCP BBR flows and balances congestion in different scenarios.

Fig. 10 shows how the dynamic network conditions affect

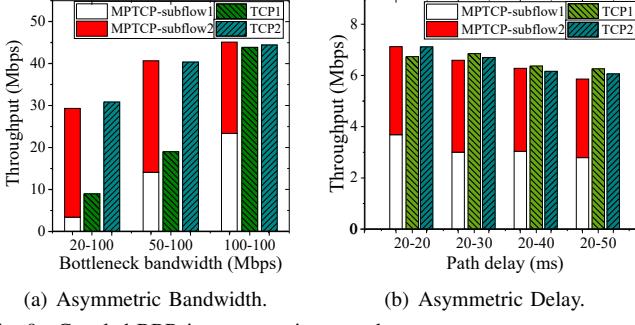


Fig. 9. Coupled BBR in asymmetric networks.

the performance of different schedulers during the transmission. As shown in Fig. 10(a), in the first 15 seconds, both the paths have high bandwidth and low RTT and no packet loss occurs. AR-Scheduling finds that both paths are in good condition and their bandwidth should be aggregated for higher goodput. As a result, AR-Scheduling decides that the two subflows should both send non-redundant packets. Meanwhile, Redundant scheduler keeps sending redundant packets which results in lower goodput. At the moment of 15 seconds, one path breaks down. The throughput of Round-Robin and AR&P drops from 40 Mbps to about 15 Mbps while redundant scheduler protects its throughput from a high packet loss rate by sending redundant packets. Although the goodput of AR&P also drops, it recovers quickly because AR-Scheduling adaptively starts sending redundant packets on the subflow with bad path conditions and packet loss does not degrade the overall goodput. By this proactive action, AR&P recovers much faster than Round-Robin when the path failure suddenly occurs, while also retains higher goodput than Redundant when subflows have good path conditions.

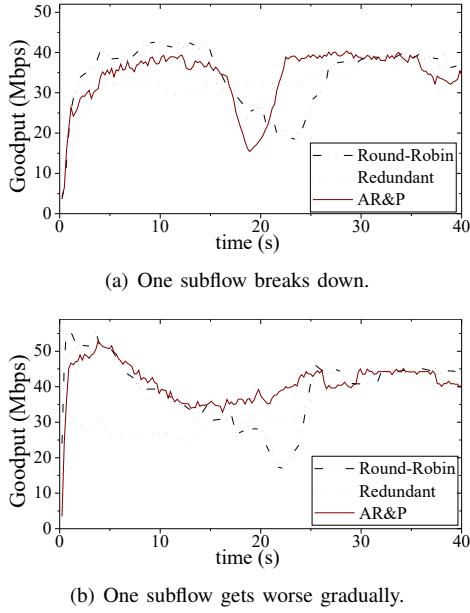


Fig. 10. Performance of proposed scheduler when changing the path loss rate and delay.

Fig. 10(b) shows another scenario in which the path conditions of one path gets worse and worse for a relatively long

period until it becomes unavailable. Round-Robin and AR&P aggregate bandwidth and outperform Redundant because both paths are in good condition at the beginning. When one of the paths gets worse and worse, the goodput of connections using Round-Robin and AR&P starts to drop. AR-Scheduling realizes that one of the paths is no longer satisfactory and starts to send redundant packets on it for better performance at 12 s, while Round-Robin keeps sending new packets resulting in a significant throughput decrease. Besides, Redundant is not affected by the path failure. During the whole transmission, AR&P scheduler is more adaptive to dynamic networks by adjusting its policy according to path conditions.

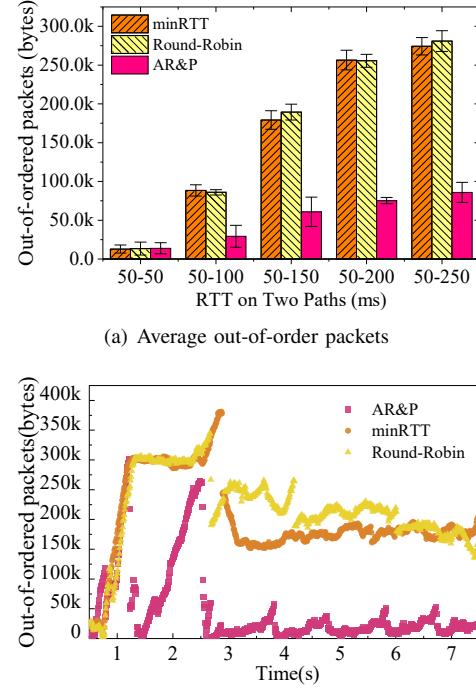


Fig. 11. Out-of-order packets.

Fig. 11 shows the out-of-order packets in asymmetric network scenarios. We compare minRTT, and Round-Robin with AR&P scheduler. In this experiment, both the bottlenecks have the same bandwidth. RTT of subflow<sub>1</sub> remains 50 ms, while RTT of the other one increases from 50 ms to 250 ms in different scenarios. When the two paths have the same RTT of 50 ms, the proposed scheduler creates a similar out-of-order queue to minRTT and Round-Robin. However, when the RTT of one path reaches 100 ms, we observe that both minRTT and Round-Robin increase out-of-order queues by over 300%, which is much longer than that of AR&P scheduler. When the RTT of one path reaches 250 ms, which means that the two paths are highly asymmetric in terms of RTT, AR&P scheduler reduces the average out-of-order queue by 65% compared to minRTT and Round-Robin.

To look further, Fig. 11(b) shows how the out-of-order queues change during data transmission when the RTT of the two paths is 150 ms and 50 ms, respectively. In the first 2 seconds, all of the schedulers create long out-of-order queues because of startup and asynchronous subflow establishment.

After 2 seconds, AR&P scheduler keeps the out-of-order queue much shorter than minRTT and Round-Robin. We observe that our scheduler empties the out-of-order queue before it gets too long, which indicates that our scheduler effectively schedules packets according to the arrival time of each packet. However, minRTT and Round-Robin are not aware of the arrival time of packets and thus create long out-of-order queues.

### B. Experiments in Real Networks

We also deploy Linux kernels that support our scheme in cloud servers to conduct some tests in real networks, transmitting data from the implemented server in the cloud to the lab-built client. We use different kinds of Wi-Fi links (2.4GHz and 5GHz) and deploy our scheme in the rented cloud servers in different regions to conduct some experiments. We repeat 10-20 times of data download under different network environments (10 for using MPTCP flows and 20 for using TCP flows on different links). When we use the proposed MPTCP, the compared TCP flow uses BBR. Otherwise, the compared TCP flow uses NewReno.

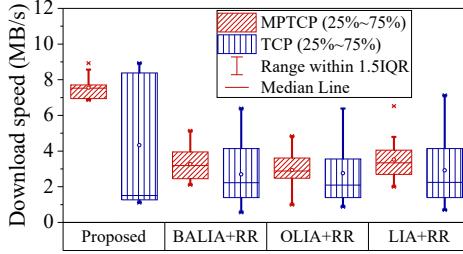


Fig. 12. Download data using 4G and Wi-Fi (2.4GHz).

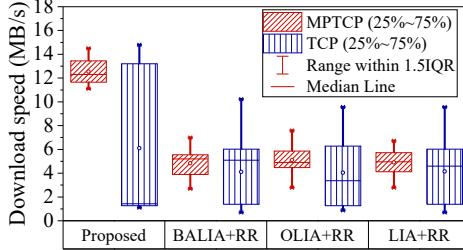


Fig. 13. Download data using 4G and Wi-Fi (5GHz).

We first show the performance measurements of using different access technologies. Fig. 12 shows the throughput performance using 4G and Wi-Fi (2.4GHz). In our test environment, the bandwidth of the Wi-Fi (2.4GHz) link is twice as fast as the 4G link. Moreover, the 4G link has a higher link packet loss rate, which makes the transmission not as stable as the Wi-Fi link. The boxes show the 25%-75% of the download speed of each protocol and the lines show the median download speed. MPTCP flows always have a higher average speed than TCP flows, while also provides less fluctuation of performance. Among them, the proposed MPTCP scheme outperforms original MPTCP algorithms in higher throughput. The overall throughput of the proposed scheme is twice higher than that of the original MPTCP. At the same time, the proposed scheme also achieves the goal of

fairness, i.e., the proposed MPTCP flow is no more aggressive than the best single TCP BBR flow.

Fig. 13 shows the download speed using 4G and Wi-Fi (5GHz). 5GHz Wi-Fi link has higher bandwidth but is not as stable as the 2.4GHz link which has a higher random loss rate. Compared with the original MPTCP, the proposed scheme brings more advantages in this scenario. The throughput of our scheme is almost 3 times higher than that of original MPTCP algorithms.

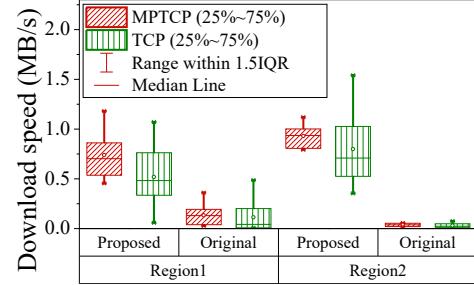


Fig. 14. Download data from different regions.

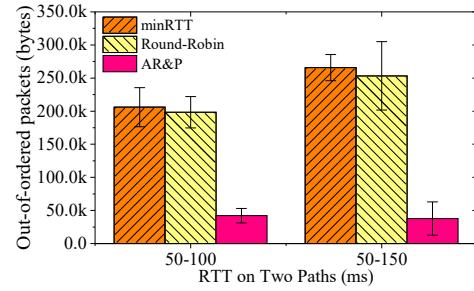


Fig. 15. Average out-of-order packets in the real networks.

Moreover, we deploy our scheme on cloud servers in several regions, where the paths suffer large RTT and random loss rate. In this case, our scheme is more outstanding than others. Fig. 14 shows the performance result. In this scenario, the throughput of the original MPTCP is less than 0.2MB/s, which is far less than the available bandwidth of devices' interfaces. This is because large packet loss hinders the growth of the congestion window, and the packets in the small congestion window suffer from large RTT transmitted to the receiver. However, wherever the server is, MPTCP with our scheme achieves throughput over 10 times higher than that of original MPTCP, showing the superiority of our scheme in the networks with bad conditions.

Fig. 15 shows the average out-of-order packets in the real networks. MPTCP server is deployed in two cloud MPTCP servers of different regions. Our client establishes two subflows through which the two servers access 4G and Wi-Fi, respectively, and the RTT of the subflows using the two accesses are shown in Fig. 15. In this experiment, our AR&P scheduler keeps the out-of-order queue short, while minRTT and Round-Robin schedulers create up to 5 times longer out-of-order queue than AR&P does. When the difference between the two subflows is getting larger, AR&P does not create a longer out-of-order queue while the other two schedulers do create more out-of-order packets.

In summary, Coupled BBR and AR&P Scheduler make MPTCP more feasible in real networks. With our proposed schemes, MPTCP throughput can be improved by up to 2.5 times in normal wireless scenarios and more than 10 times in other scenarios with large RTT and loss. Moreover, the number of out-of-order packets can be reduced by 80% at most in asymmetric scenarios.

### C. Performance Simulation

We utilize a network topology shown in Fig. 16 to test the performance of Coupled BBR and AR&P Scheduler. The MPTCP connection includes three subflows, each of which passes through a path with bandwidth of  $B_i$  on the bottleneck. There is one TCP flow that passes through the same path of each subflow. The link delay and random packet loss rate are set to  $d_i$  and  $p_i$  of each path  $i$ .

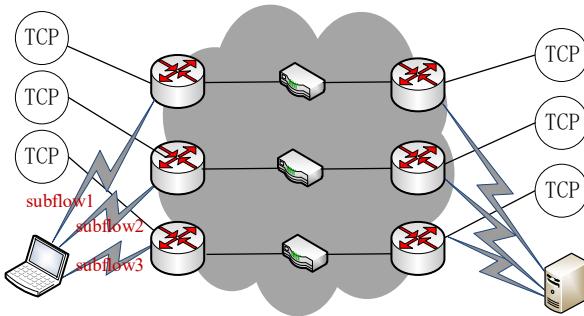


Fig. 16. Topology of simulation.

By utilizing the bottleneck bandwidth detection method, Coupled BBR provides high bandwidth utilization, especially in lossy and long-delay networks. As shown in Fig. 17, we compare Coupled BBR with LIA, BALIA, and OLIA in different network scenarios. TCP BBR, TCP newReno (B), (O), (L) are the background TCP flow on the MPTCP Coupled BBR, BALIA, OLIA, LIA, respectively. We set  $B_1 = B_2 = B_3 = 100$  Mbps,  $d_1 = d_2 = d_3 = 20$  ms and  $p_1 = p_2 = p_3 = 0 - 0.5\%$ , respectively. The overall bandwidth resource of the network is 300 Mbps. When the random loss rate is 0, both Coupled BBR and other MPTCP congestion control algorithms can achieve high bandwidth utilization. With the increase of random loss rate, the throughput of Coupled BBR almost does not decline. The throughput of other MPTCP congestion control algorithms declines significantly. In Fig. 17(b), we set  $B_1 = B_2 = B_3 = 100$  Mbps,  $d_1 = d_2 = d_3 = 2 - 100$  ms and  $p_1 = p_2 = p_3 = 0.05\%$ , respectively. When the path delay is low, the link random packet loss does not make a great effect on the original congestion algorithms. With the increase of path delay, the throughput of LIA, OLIA, BALIA declines significantly, while Coupled BBR still achieves high throughput and bandwidth utilization.

Fig. 18 shows the performance of Coupled BBR in ever-changing networks. We set  $B_1 = B_2 = B_3 = 100$  Mbps,  $d_1 = d_2 = d_3 = 20$  ms and  $p_1 = p_2 = p_3 = 0.01\%$  at beginning of the transmission. During the transmission,  $p_1$ ,  $p_2$  and  $p_3$  linearly change to 1% from 10 s to 30 s. With the increase of

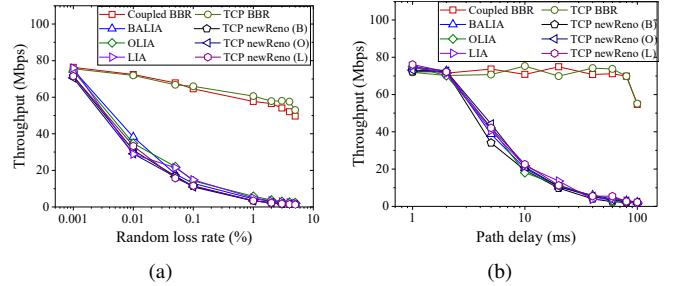


Fig. 17. Performance in lossy network scenarios, (a) Random loss rate changes, (b) Path delay changes.

path loss rate, the throughput of BALIA, OLIA, LIA decreases significantly. Compare with them, Coupled BBR has always maintained high throughput.

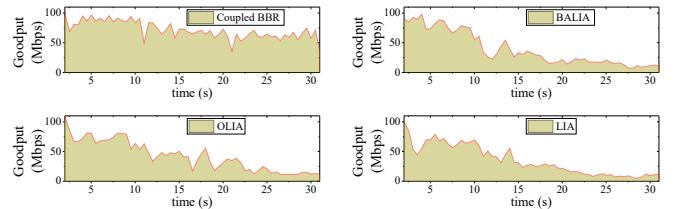


Fig. 18. Real-time throughput of different algorithms in an ever-changing lossy network scenario.

Fig. 19 shows the performance of AR&P Scheduler in various networks. We change  $B_1, B_2, B_3$  from 10 to 100 Mbps,  $d_1, d_2, d_3$  from 1 to 100 ms and  $p_1, p_2, p_3$  from 0 to 5%. We randomly pick up 100 points in the parameter space for simulation and analyze the simulation results. As shown in Fig. 19(a), the throughput of Coupled BBR with both minRTT and AR&P is much higher than that of BALIA, OLIA, and LIA in random loss scenarios. Among them, AR&P further improves the throughput based on Coupled BBR. Compare with BALIA, OLIA and LIA using the same scheduler minRTT, Coupled BBR with minRTT has much higher out-of-order packets. Because the number of out-of-order packets is related to throughput and Coupled BBR significantly improves the throughput. AR&P scheduler further improves the goodput and reduce the number of out-of-order packets on the basis of Coupled BBR. Compare with BALIA, OLIA and LIA with minRTT, Coupled BBR with AR&P gains much higher goodput with the same low-level out-of-order packets, which can be seen in Fig. 19(b).

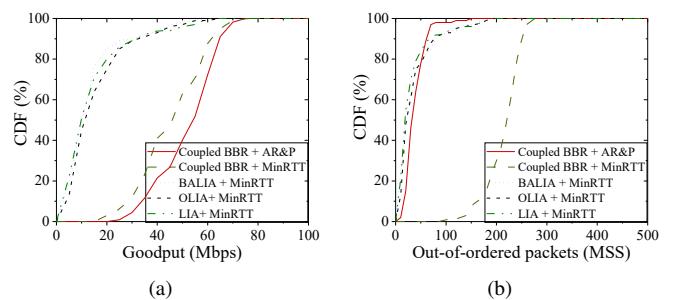


Fig. 19. Performance in various network scenarios, (a) Average goodput, (b) Average out-of-ordered packets.

Fig. 20 shows the performance of AR&P Scheduler in an ever-changing network, where OFO packets denote out-of-order packets. We set  $B_1 = B_2 = B_3 = 100$  Mbps,  $d_1 = d_2 = d_3 = 20$  ms and  $p_1 = p_2 = p_3 = 0.01\%$  at beginning of the transmission. During the transmission,  $B_2$ ,  $d_2$  and  $p_2$  change to 10 Mbps, 100 ms and 1% from 10 s to 15 s.  $B_3$ ,  $d_3$  and  $p_3$  change to 10 Mbps, 100 ms and 1% from 20 s to 25 s. BALIA, OLIA, and LIA with minRTT all experience low goodput when one of the paths is changing. When the delay of one of the paths increases (after 10 s), their out-of-ordered packets also increase significantly. Compare with them, Coupled BBR with minRTT improves the goodput, while the number of out-of-ordered packets also increases due to the high throughput. Coupled BBR with AR&P further promotes the goodput and significantly reduce out-of-ordered packets when path conditions change. When the path conditions begin to change, P-Scheduling method helps to schedule packets and reduces out-of-ordered packets. When the path condition continues to deteriorate, AR-Scheduling decides to send redundant packets on the subflow with bad path conditions. Therefore the out-of-ordered packets are further reduced.

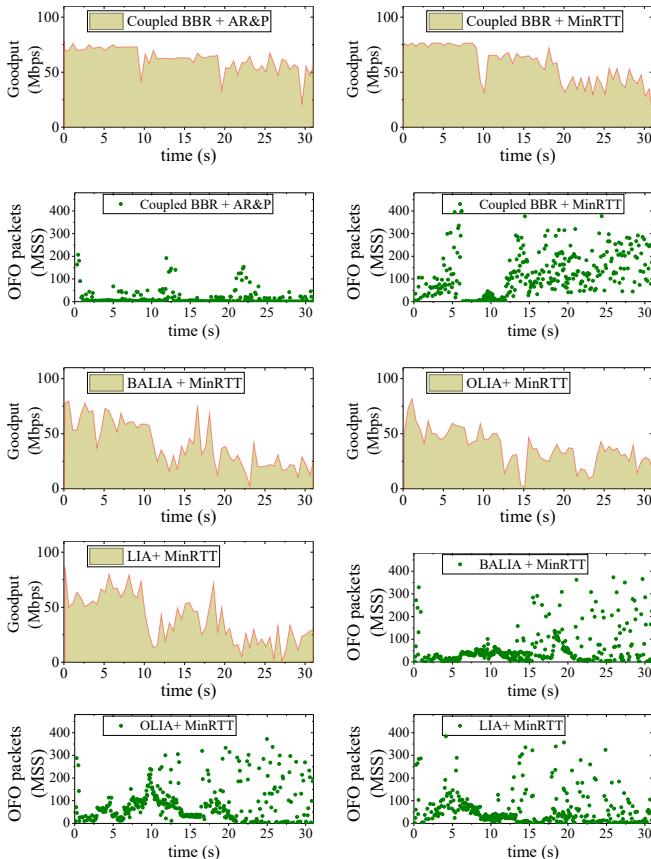


Fig. 20. Real-time goodput and out-of-ordered packets of different algorithms in an ever-changing network scenario.

## VII. RELATED WORK

**Coupled congestion control algorithms:** The basic goal of coupled congestion control algorithms in MPTCP is to achieve fairness with TCP flows, but it also needs to further achieve

additional goals such as congestion balance. Current coupled congestion control algorithms, such as LIA [7], OLIA [32], and BALIA [33] couple the congestion control algorithms on different subflows by linking their increase function in AIMD based on TCP NewReno. For every RTT on subflow $_i$ , coupled congestion control algorithms increase the congestion window  $w_i$  by a parameter  $\alpha_i$  instead of 1 in NewReno. Thus in the network with a certain loss rate, different speeds of window increasing will lead to different overall throughput. However, as BBR does not include the AIMD method, any AIMD-based scheme is not suitable for developing MPTCP over BBR congestion control.

**Scheduling algorithms:** Scheduling algorithms are mainly designed for improving robustness, reducing the out-of-order packets, or reducing latency. Based on the operating patterns, they can be divided into several categories: 1) Simple schedulers in Linux Kernel [26], which are Round-Robin, minRTT, and Redundant. Round-Robin polls subflows and sends packets in order. minRTT always sends packets on the available subflow with the lowest RTT. Redundant sends redundant packets to ensure high robustness and low latency. 2) Schedulers acting on paths. This kind of scheduler improves MPTCP performance by controlling each path's action [14], [15], [34]. For example, Musher [14] controls the allocation rate of data on each path to get better throughput. RAVEN [15] mitigates tail latency by using redundant transmission when confidence about network latency predictions is low. 3) Schedulers acting on packets [4], [35]–[37]. These proposed schedulers, like ECF [36], STMS [4], STTF [37], aim at keeping low latency and reducing out-of-order packets in asymmetric networks. They schedule packets with the larger sequence number to the subflow with larger RTT so as to keep packets delivery in order. However, existing scheduling algorithms are based on traditional congestion control algorithms, which in turn depends on the congestion window, and thus do not work for Coupled BBR.

## VIII. DISCUSSION

In this work, we study the fairness between MPTCP Coupled BBR flows and TCP BBR flows. We consider a network that uses BBR to control all the flows so that the network is more stable and all the flows can get better performance. A full BBR network provides more advantages for developing higher performance transmission protocols in the future. Moreover, in lossy networks, traditional loss-based congestion control algorithms could not make good use of the available bandwidth of the bottleneck, which makes it meaningless to achieve fairness between BBR and other algorithms in this case. Moreover, we address network fairness. To be noted that, for bottleneck fairness, MPTCP subflows sharing one bottleneck should be coupled to achieve fairness with TCP flows in the same bottleneck, and it just needs a bottleneck detection method for Coupled BBR. Then our scheme can be easily adapted for it.

## IX. CONCLUSION

In this work, we propose Coupled BBR and AR&P scheduler to improve the performance of MPTCP in lossy or

ever-changing networks. With Coupled BBR, MPTCP not only performs well in lossy circumstances but also balances congestion among subflows and achieves fairness to TCP BBR flows. AR&P scheduler further enhances MPTCP performance in dynamic and asymmetric networks with two scheduling methods to provide better self-adaptability and reduce the out-of-order packets.

## ACKNOWLEDGMENT

The work of Jiangping Han, Kaiping Xue, Yitao Xing, Jian Li and Wenjia Wei was supported in part by the National Natural Science Foundation of China (NSFC) under Grant No. 61972371, and Youth Innovation Promotion Association of the Chinese Academy of Sciences (CAS) under Grant No. 2016394. The work of Guoliang Xue was supported in part by National Science Foundation (NSF) under Grant No. 1704092. The information reported here does not reflect the position or the policy of the funding agencies.

## REFERENCES

- [1] A. Ford, C. Raiciu, M. J. Handley, and O. Bonaventure, “TCP extensions for multipath operation with multiple addresses,” RFC 6824, Jan. 2013. [Online]. Available: <https://rfc-editor.org/rfc/rfc6824.txt>
- [2] C. Raiciu, S. Barre, C. Pluntke, A. Greenhalgh, D. Wischik, and M. Handley, “Improving datacenter performance and robustness with multipath TCP,” in *ACM SIGCOMM Computer Communication Review*, vol. 41, no. 4. ACM, 2011, pp. 266–277.
- [3] M. Li, A. Lukyanenko, Z. Ou, A. Ylä-Jääski, S. Tarkoma, M. Coudron, and S. Secci, “Multipath transmission for the internet: a survey,” *IEEE Communications Surveys & Tutorials*, vol. 18, no. 4, pp. 2887–2925, 2016.
- [4] H. Shi, Y. Cui, X. Wang, Y. Hu, M. Dai, F. Wang, and K. Zheng, “STMS: Improving MPTCP throughput under heterogeneous networks,” in *Proceedings of 2018 USENIX Annual Technical Conference (ATC)*, 2018, pp. 719–730.
- [5] O. Bonaventure, C. Paasch, and G. Detal, “Use cases and operational experience with Multipath TCP,” RFC 8041, Jan. 2017. [Online]. Available: <https://rfc-editor.org/rfc/rfc8041.txt>
- [6] T. Zhang, S. Zhao, B. Cheng, B. Ren, and J. Chen, “Research, implementation, and improvement of MPTCP on mobile smart devices,” *International Journal of Computers and Applications*, pp. 1–11, 2018.
- [7] D. Wischik, C. Raiciu, A. Greenhalgh, and M. Handley, “Design, implementation and evaluation of congestion control for multipath TCP,” in *Proceedings of the 8th USENIX conference on Networked systems design and implementation (NSDI)*, 2011.
- [8] C. Raiciu, M. J. Handley, and D. Wischik, “Coupled congestion control for multipath transport protocols,” RFC 6356, Oct. 2011. [Online]. Available: <https://rfc-editor.org/rfc/rfc6356.txt>
- [9] S. Ferlin, Ö. Alay, T. Dreibholz, D. A. Hayes, and M. Welzl, “Revisiting congestion control for multipath TCP with shared bottleneck detection,” in *Proceeding of the 35th IEEE Annual IEEE International Conference on Computer Communications (INFOCOM)*. IEEE, 2016, pp. 1–9.
- [10] C. Xu, J. Zhao, and G.-M. Muntean, “Congestion control design for multipath transport protocols: a survey,” *IEEE Communications Surveys & Tutorials*, vol. 18, no. 4, pp. 2948–2969, 2016.
- [11] H. Sinky, B. Hamdaoui, and M. Guizani, “Proactive multipath TCP for seamless handoff in heterogeneous wireless access networks,” *IEEE Transactions on Wireless Communications*, vol. 15, no. 7, pp. 4754–4764, 2016.
- [12] K. W. Choi, Y. S. Cho, J. W. Lee, S. M. Cho, J. Choi *et al.*, “Optimal load balancing scheduler for MPTCP-based bandwidth aggregation in heterogeneous wireless environments,” *Computer Communications*, vol. 112, pp. 116–130, 2017.
- [13] Y. E. Guo, A. Nikravesh, Z. M. Mao, F. Qian, and S. Sen, “Accelerating multipath transport through balanced subflow completion,” in *Proceedings of the 23rd Annual International Conference on Mobile Computing and Networking (MobiCom)*. ACM, 2017, pp. 141–153.
- [14] S. K. Saha, S. Aggarwal, R. Pathak, D. Koutsonikolas, and J. Widmer, “MuSher: An agile Multipath-TCP scheduler for dual-band 802.11 ad/ac wireless LANs,” in *Proceedings of the 25th Annual International Conference on Mobile Computing and Networking (MobiCom)*. ACM, 2019, p. 34.
- [15] H. Lee, J. Flinn, and B. Tonshol, “RAVEN: Improving interactive latency for the connected car,” in *Proceedings of the 24th Annual International Conference on Mobile Computing and Networking (MobiCom)*. ACM, 2018, pp. 557–572.
- [16] A. Nikravesh, Y. Guo, F. Qian, Z. M. Mao, and S. Sen, “An in-depth understanding of multipath TCP on mobile devices: measurement and system design,” in *Proceedings of the 22nd Annual International Conference on Mobile Computing and Networking (MobiCom)*. ACM, 2016, pp. 189–201.
- [17] S. R. Pokhrel, M. Panda, and H. L. Vu, “Analytical modeling of multipath TCP over last-mile wireless,” *IEEE/ACM Transactions on Networking (ToN)*, vol. 25, no. 3, pp. 1876–1891, 2017.
- [18] S. Ferlin, S. Kucera, H. Claussen, and O. Alay, “MPTCP meets FEC: Supporting latency-sensitive applications over heterogeneous networks,” *IEEE/ACM Transactions on Networking (ToN)*, vol. 26, no. 5, pp. 2005–2018, 2018.
- [19] L. Li, K. Xu, T. Li, K. Zheng, C. Peng, D. Wang, X. Wang, M. Shen, and R. Mijumbi, “A measurement study on multi-path TCP with multiple cellular carriers on high speed rails,” in *Proceedings of the 2018 Conference of the ACM Special Interest Group on Data Communication (SIGCOMM)*. ACM, 2018, pp. 161–175.
- [20] A. Kumar, “Comparative performance analysis of versions of TCP in a local network with a lossy link,” *IEEE/ACM Transactions on Networking (ToN)*, vol. 6, no. 4, pp. 485–498, 1998.
- [21] S. Ha, I. Rhee, and L. Xu, “CUBIC: a new TCP-friendly high-speed TCP variant,” *ACM SIGOPS operating systems review*, vol. 42, no. 5, pp. 64–74, 2008.
- [22] Y. Cao, A. Jain, K. Sharma, A. Balasubramanian, and A. Gandhi, “When to use and when not to use BBR: An empirical analysis and evaluation study,” in *Proceedings of the Internet Measurement Conference*. ACM, 2019, pp. 130–136.
- [23] W. Wei, K. Xue, J. Han, D. S. Wei, and P. Hong, “Shared Bottleneck-Based congestion control and packet scheduling for multipath TCP,” *IEEE/ACM Transactions on Networking*, vol. 28, no. 2, pp. 653–666, 2020.
- [24] Y. Thomas, M. Karaliopoulos, G. Xylomenos, and G. C. Polyzos, “Low latency friendliness for multipath TCP,” *IEEE/ACM Transactions on Networking*, vol. 28, no. 1, pp. 248–261, 2020.
- [25] Y. R. Yang and S. S. Lam, “General AIMD congestion control,” in *Proceedings of International Conference on Network Protocols (ICNP)*. IEEE, 2000, p. 187.
- [26] “MultiPath TCP - linux kernel implementation,” <http://www.multipath-tcp.org/>.
- [27] A. A. Abouzeid, S. Roy, and M. Azizoglu, “Comprehensive performance analysis of a TCP session over a wireless fading link with queueing,” *IEEE Transactions on wireless Communications*, vol. 2, no. 2, pp. 344–356, 2003.
- [28] M. Zhang, M. Polese, M. Mezzavilla, J. Zhu, S. Rangan, S. Panwar, and M. Zorzi, “Will TCP work in mmWave 5g cellular networks?” *IEEE Communications Magazine*, vol. 57, no. 1, pp. 65–71, 2019.
- [29] J. P. Singh, Y. Li, N. Bamboz, A. Bahai, B. Xu, and G. Zimmermann, “TCP performance dynamics and link-layer adaptation based optimization methods for wireless networks,” *IEEE Transactions on Wireless Communications*, vol. 6, no. 5, pp. 1864–1879, 2007.
- [30] B.-H. Oh and J. Lee, “Feedback-based path failure detection and buffer blocking protection for MPTCP,” *IEEE/ACM Transactions on Networking (ToN)*, vol. 24, no. 6, pp. 3450–3461, 2016.
- [31] N. Cardwell, Y. Cheng, S. H. Yeganeh, and V. Jacobson, “BBR congestion control,” *IETF Secretariat, Internet-Draft draft-card-well-icrg-bbr-congestion-control-00*, 2017.
- [32] R. Khalili, N. Gast, M. Popovic, and J.-Y. Le Boudec, “MPTCP is not pareto-optimal: performance issues and a possible solution,” *IEEE/ACM Transactions on Networking (ToN)*, vol. 21, no. 5, pp. 1651–1665, 2013.
- [33] Q. Peng, A. Walid, J. Hwang, and S. H. Low, “Multipath TCP: Analysis, design, and implementation,” *IEEE/ACM Transactions on Networking (ToN)*, vol. 24, no. 1, pp. 596–609, 2016.
- [34] B. Han, F. Qian, L. Ji, and V. Gopalakrishnan, “MP-DASH: Adaptive video streaming over preference-aware multipath,” in *Proceedings of the 12th International on Conference on emerging Networking EXperiments and Technologies*. ACM, 2016, pp. 129–143.
- [35] K. Xue, J. Han, D. Ni, W. Wei, Y. Cai, Q. Xu, and P. Hong, “DPSAF: Forward prediction based dynamic packet scheduling and adjusting with

feedback for multipath TCP in lossy heterogeneous networks," *IEEE Transactions on Vehicular Technology*, vol. 67, no. 2, pp. 1521–1534, 2018.

[36] Y.-s. Lim, E. M. Nahum, D. Towsley, and R. J. Gibbens, "ECF: An MPTCP path scheduler to manage heterogeneous paths," in *Proceedings of the 13th International Conference on emerging Networking EXperiments and Technologies*. ACM, 2017, pp. 147–159.

[37] P. Hurtig, K.-J. Grinnemo, A. Brunstrom, S. Ferlin, Ö. Alay, and N. Kuhn, "Low-latency scheduling in MPTCP," *IEEE/ACM Transactions on Networking (ToN)*, vol. 27, no. 1, pp. 302–315, 2018.



**Jian Li** received his B.S. degree from the Department of Electronics and Information Engineering, Anhui University, in 2015, and received Ph.D degree from the Department of Electronic Engineering and Information Science (EEIS), University of Science and Technology of China (USTC), in 2020. From Nov. 2019 to Nov. 2020, he was a visiting scholar with the Department of Electronic and Computer Engineering, University of Florida. He is currently a Post-Doctoral researcher with the Department of EEIS, USTC. His research interests include wireless communications, satellite networks and next-generation Internet.



**Jiangping Han** receives her B.S. degree from the Department of Electronic Engineering and Information Science (EEIS), USTC, in July, 2016. She is currently working toward the Ph.D degree in communication and information systems with also the Department of EEIS, USTC. Her research interests include future Internet architecture design and transmission optimization. Part of this work was done while she was a visiting student at Arizona State University.



**Wenjia Wei** received the B.S. degree from the school of infomation science and engineering, in 2013. He received the Ph.D. degree in Information and Communication Engineering from the Department of Electronic Engineering and Information Science (EEIS), University of Science and Technology of China (USTC), in 2020. His research interests include future Internet architecture design and transmission optimization.



**Kaiping Xue** (M'09-SM'15) received his bachelor's degree from the Department of Information Security, University of Science and Technology of China (USTC), in 2003 and received his 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 the Department of Electrical and Computer Engineering, University of Florida. Currently, he is a Professor in the School of Cyber Security and the Department of EEIS, USTC. His research interests include next-generation Internet, distributed networks and network security. He serves 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 has also served as a guest editor of IEEE Journal on Selected Areas in Communications (JSAC) and a lead guest editor of IEEE Communications Magazine. He is an IET Fellow and an IEEE Senior Member. He is the corresponding author of this paper.



**David S.L. Wei** (SM'07) received his 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 at the University of Aizu, Japan (as an Associate Professor and then a Professor). He has authored and co-authored more than 120 technical papers in various archival journals and conference proceedings. He is currently a Professor of Computer and Information Science Department at 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 IEEE Transactions on Big Data. He also served as an Associate Editor of IEEE Transactions on Cloud Computing, 2014-2018, and an Associate Editor of Journal of Circuits, Systems and Computers, 2013-2018.



**Yitao Xing** received his B.S. degree in Information Security from 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 from the Department of Electronic Engineering and Information Science (EEIS), USTC. His research interests include future Internet architecture and transmission optimization.



**Guoliang Xue** (M'96-SM'99-F'11) is a professor of Computer Science and Engineering at Arizona State University. He received the Ph.D degree in Computer Science from the University of Minnesota in 1991. His research interests span the areas of QoS provisioning, machine learning, wireless networking, and IoT. He has received the IEEE Communications Society William R. Bennett Prize in 2019. He has served as a TPC Chair of IEEE INFOCOM'2010, IEEE Globecom'2020, and IEEE IWQoS'2021, and a General Chair of IEEE CNS'2014 and IEEE/ACM IWQoS'2019. He has served on the TPC of many conferences, including ACM CCS, ACM MOBIHOC, IEEE ICNP, and IEEE INFOCOM. He served on the editorial board of IEEE/ACM Transactions on Networking and the Area Editor of IEEE Transactions on Wireless Communications, overseeing 12 editors in the Wireless Networking area. He is an IEEE Fellow, and the Steering Committee Chair of IEEE INFOCOM.