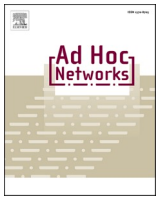
Ad Hoc Networks 162 (2024) 103560

Contents lists available at ScienceDirect

Ad Hoc Networks

journal homepage: www.elsevier.com/locate/adhoc



Enhancing MPTCP performance in wireless networks with PRS-MPTCP

Atefeh Ahmadniai Khajekini [a,*], Hasan Amca [a], Ali Hakan Ulusoy [b], Enver Ever [c]

[a] Department of Electrical and Electronic Engineering, Faculty of Engineering, Eastern Mediterranean University, Famagusta, North Cyprus, via Mersin 10, Turkey
[b] Department of Information Technology, School of Computing and Technology, Eastern Mediterranean University, Famagusta, North Cyprus, via Mersin 10, Turkey
[c] Computer Engineering, Middle East Technical University Northern Cyprus Campus, 99738, Mersin 10, Turkey

ARTICLE INFO

Keywords:
Wireless networks
MPTCP technology
Scheduling algorithms
Mininet-WiFi
TCP method
IoT systems
M2M networks

ABSTRACT

In recent years, there has been a significant increasing demand for wireless networks, particularly mobile communication. Additionally, smart devices like mobile phones and tablets are able to use multiple interfaces simultaneously. In this regard, the concept of Multipath TCP (MPTCP) has been introduced, enabling the utilization of multiple interfaces for concurrent communication. However, packet loss and subflow heterogeneity, especially in wireless networks, cause an increase in Out-of-Order (OfO) packets at the receiver node, which leads to a decrease in the total MPTCP throughput. To address these performance-related challenges, numerous schedulers have been proposed. However, most existing methods have primarily focused on improving performance without adequately considering the impact of packet loss. This research paper provides a comprehensive overview of MPTCP schedulers. Subsequently, we propose a Practical and Robust Scheduler for MPTCP (PRS-MPTCP) with the specific aim of minimizing OfO packets to improve MPTCP performance in wireless systems. The PRS-MPTCP scheduler takes into account various characteristics of each subflow, including RTT, CWND, and the number of packet losses, to effectively decide which packets should be assigned to which subflow. By using Mininet-WiFi emulation, fair comparisons between PRS-MPTCP and the state-of-the-art schedulers have been conducted, considering throughput, the number of OfO packets, and retransmission rates as performance metrics. The evaluation results reveal the profound impact of selected parameters on the behavior of the schedulers and the overall performance of MPTCP. Ultimately, the results demonstrate that PRS-MPTCP guarantees acceptable throughput and achieves lower retransmission rates and fewer OfO packets compared to other methods. In the PRS-MPTCP, the number of OfO packets has decreased by 38 %, 37 %, 45 %, and 44 % compared to BLEST, ECF, RR, and the Default scheduler, respectively.

1. Introduction

In today's world, mobile systems have emerged as the predominant communication networks, playing a central role in global connectivity. In addition, because of modern technologies such as Machine-to-Machine (M2M) communications and the Internet of Things (IoT), the number of smart devices exceeds the user's population [5]. Both wired and wireless networks support mobile devices (e.g., smartphones). The involved devices can employ multi-wireless communication interfaces to connect to a broad range of networks like cellular (e.g., 3 G/4 G/LTE/5 G) and local area networks (e.g., Wi-Fi). Employing multiple interfaces improves network access, which is truly ubiquitous to handle IoT and M2M systems appropriately [1–5]. With the quick development of network technologies, especially wireless networks, many more wireless devices are equipped with multiple interfaces to work with heterogeneous networks, including Wireless Local Area Networks (WLAN) and cellular networks. In addition, because of the increasing number of multi-homed mobile devices, which need more access networks, the M2M devices can switch between different access networks based on Quality of Service (QoS) metrics. With more devices in IoT systems, the request for packet transmission through mobile systems has increased considerably, requiring some practical protocols in communication systems.

The main existing protocols, like TCP/IP, can use only one interface for each connection. Thus, TCP cannot employ all available interfaces efficiently. This limits the performance of multi-homed mobile devices, in particular when we have a non-stable network since using more than one network interface is not present in the common TCP scheme. The TCP method employs only one single path to send and receive packets in the access networks [6].

* Corresponding author.
E-mail address: atefeh.ahmadniai@emu.edu.tr (A.A. Khajekini).

https://doi.org/10.1016/j.adhoc.2024.103560
Received 26 January 2024; Received in revised form 2 April 2024; Accepted 21 May 2024
Available online 23 May 2024

The utilization of network resources can be maximized through the utilization of Multipath TCP (MPTCP), which serves as an effective method for enhancing the performance of IoT and M2M systems. This is achieved by simultaneously utilizing all accessible subflows [7,8]. Accordingly, MPTCP is an extension of the original TCP to provide an enhanced transmission between two end devices (e.g., client and server) over multiple interfaces (i.e., paths). The MPTCP is able to send and receive data using all interfaces between the server node and the client node. On the client side, user's smartphones can connect to the Wi-Fi network and LTE system simultaneously and send/receive data through both interfaces. On the server side, different servers (data centers) can be connected by means of several wired/wireless paths to increase the level of fault tolerance in the system [9]. The MPTCP provides this feature on both the client and server-side to enhance the level of QoS in the system. MPTCP can provide concurrent data transmission by means of employing multiple paths, which are named subflows, to improve the throughput of the network. Using several paths, MPTCP can achieve much better QoS than common TCP [10–14]. In the MPTCP, each subflow is determined by emerging technologies to modify data forwarding based on different network conditions. Therefore, the reliability of the system against link failure can be improved significantly. Thus, compared with single-path TCP, MPTCP technology can provide important resilience to failure and better performance in communication systems [15,16].

The TCP is originally designed for wired access networks, while the MPTCP is designed for wireless networks, in particular for IoT applications. The implementations of MPTCP in different distributions of Linux Kernel are the most common implementations of MPTCP for M2M systems. It is also used in iOS and Giga Path in Korean Telecom [12]. Han et al. [17] apply MPTCP to improve the user experience on video streaming. The MPTC is also able to improve the performance of several commonly used systems like Live Video Streaming (LVS) applications [5]. However, with the increasing number of LVS applications and smart devices, achieving appropriate data transmission for mobile users has become a significant issue. Moreover, user's devices need to connect to the access networks continuously, and these devices must be supported by different networks (e.g., 4 G, 5 G, and Wi-Fi). There are some main requirements in the LVS applications. They are bandwidth-consuming and need high-quality capabilities. Some LVS applications require real-time transmission latency. A practicable method to solve these issues is to apply MPTCP well-appointed in the user's smart device to access two interfaces, LTE and WLAN, simultaneously [18].

There are two important points in MPTCP, which are the congestion control mechanism method and path scheduler [19–26]. The main objective of a suitable scheduler is to decrease Head of Line (HoL) blocking at the receiver node. The Congestion Control Algorithm (CCA) goal is to improve the utilization of each available subflow. The CCA is designed to determine the Congestion Window (CWND) size of each subflow and modify it after each transmission [25]. A path scheduler assigns the transmitted packets to the involved paths. If the packet scheduler works inappropriately, it may cause some performance issues not only in heterogeneous networks but also in homogeneous networks. The main performance challenges in MPTCP are high Out-of-Order (OfO) packets, computational time, low utilization and throughput [12]. Heterogeneity in MPTCP also causes issues like increased buffered packets at the receiver's node that significantly cause HoL blocking issues. On the other hand, a proper path scheduler can improve the performance of MPTCP by solving the HoL blocking issue and the receiver buffer size.

In this paper, we focus on the MPTCP scheduling mechanism in wireless systems. Each MPTCP method must specify the order of use of each path (subflow) between all available subflows. An MPTCP scheduler is responsible for making this decision. Initially, the path scheduler checks how many available subflows are in the system that can be used for data transmission using MPTCP. If the number of available paths is more than one subflow, the path scheduler can set the order of using each path according to its scheduling policy. If the scheduler method works inappropriately, the selection may cause HoL blocking challenges of receiver window limitations that can be a reason for the low QoS of the system. In addition, the MPTCP deals with multi-wireless interfaces with different characteristics. Therefore, there is a challenging issue in providing acceptable throughput of the multiple paths in MPTCP systems.

To handle the diversity of multiple paths in MPTCP, a scheduler needs to find the best available path to transmit the user's packet in each round. Accordingly, the scheduling algorithm can play a primary role in improving the performance of the MPTCP system. A developed path scheduler can employ all the available interfaces to ease OfO packets and increase the system's throughput. The need for optimized and practical MPTCP path schedulers is recognized, and several suggested methods have been presented and evaluated earlier [2–10,13–16]. Many researchers presented different algorithms for multipath communications. Most of the path schedulers are unable to beneficially achieve the main goals of MPTCP, such as decreasing latency which is one of the most important factors for sensitive applications such as IoT systems [1, 15].

We presented a set of the most widely used path schedulers in the related works section. Different MPTCP scheduling algorithms have been conducted by means of Linux Kernel implementation [27] to achieve different goals and applications. In this work, we propose a Practical and Robust Scheduler for MPTCP (PRS-MPTCP) to enhance the performance of the system efficiently. The scheduling policy of PRS-MPTCP is based on finding the best subflow between all available subflows using path characteristics, including Round-Trip Time (RTT), CWND, inflight packets, and retransmission rate. Then, we present the design of PRS-MPTCP for the requirements of Wi-Fi systems. The PRS-MPTCP scheduler can effectively utilize different interfaces. In this method, we can schedule the user's packets so as not to increase delay on the receiver side. We have checked our suggested scheduler performance based on throughput, OfO packets, and retransmission rate. The evaluation is based on different scenarios, including homogenous and heterogeneous systems.

The main contributions of this paper are listed as follows:

1. We present a comprehensive overview of MPTCP with a detailed analysis and comparison of different existing approaches under dynamic scenarios.
2. A new, novel, robust scheduler is proposed to improve the performance-based issue of MPTCP. This research paper focuses on not only the design of the suggested model but also the implementation of the scheduler in the Linux Kernel.
3. The proposed scheduler PRS-MPTCP is evaluated based on some critical performance measures, such as the number of OfO packets, the throughput of the MPTCP connection, and the retransmission rate. The proposed scheduler is compared with Blocking Estimation-based (BLEST) scheduler [10], Earliest Completion First (ECF) scheduler [13], Round Robin (RR) [1], scheduler and the Default scheduler [1] and the performance improvement in retransmission rate is presented.
4. We compare our method with the existing approaches to prove its robustness. The main reason for our robustness is that the average throughput of the PRS-MPTCP is much better than some common schedulers, including BLEST, ECF, RR, and the Default scheduler, as seen in Section 6.1.

The rest of the paper is presented as follows: Section 2 provides a description of the MPTCP background. We present the related works by considering the main existing works in Section 3. Section 4 introduces the suggested MPTCP scheduler in detail systematically. In Section 5, we present the design of the experiment and its details. In Section 6, we evaluate the proposed method and compare it with some common methods. Lastly, the conclusion section provides some key remarks

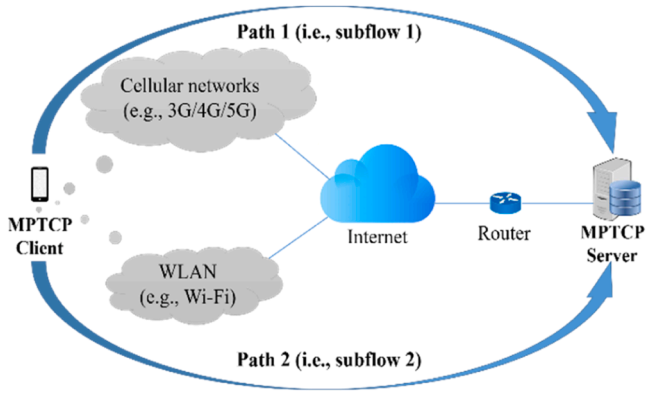


Fig. 1. MPTCP connection between client node and server node.

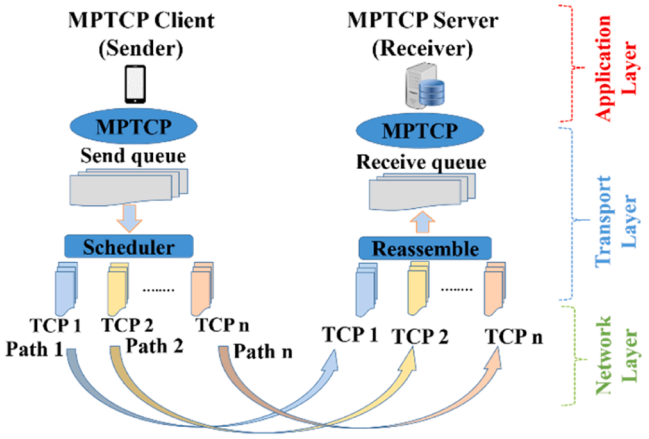


Fig. 2. Layered architecture of MPTCP in end hosts.

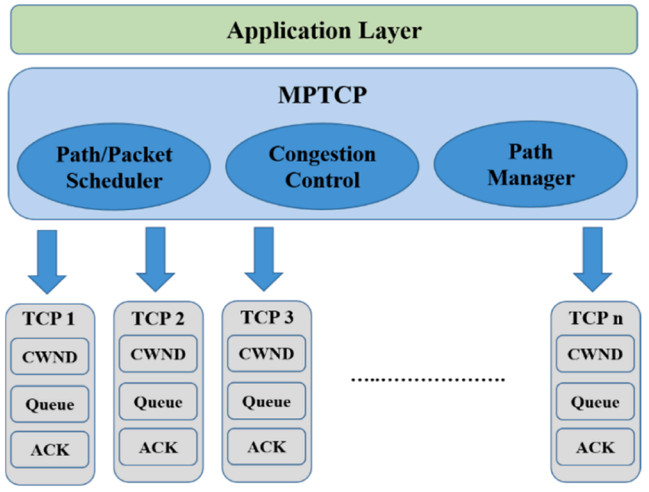


Fig. 3. MPTCP structure in a multi-homed node.

about the research achievements undertaken by our proposed method and future work.

2. Background

We discuss the main fundamentals of MPTCP architecture to accomplish a comprehensive performance analysis of the MPTCP. Mobile devices are armed with multiple network interfaces (e.g., Wi-Fi or

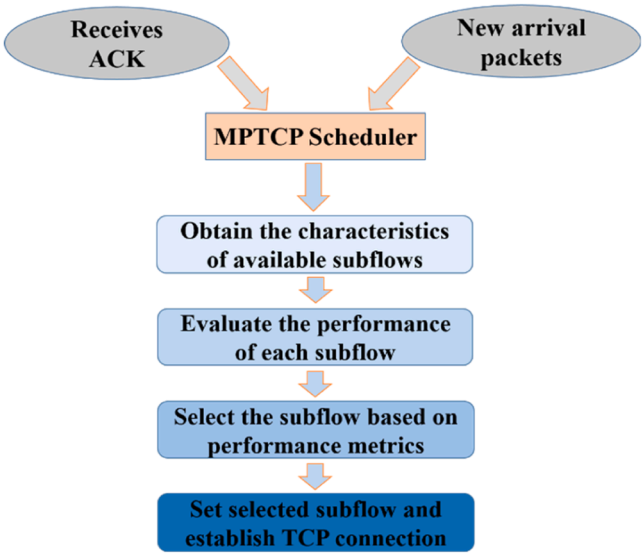


Fig. 4. Common process of MPTCP scheduler.

LTE), enabling mobile users to connect to various access networks using different wired and wireless communication channels [21]. As can be seen in Fig. 1, MPTCP can be highly effective for the client and server nodes to simultaneously use both cellular networks and WLAN for obtaining high throughput in transmission [1]. In addition, MPTCP provides continuous handover between available paths to improve network throughput and deal with link failures.

Each MPTCP connection can be configured with a common three-way handshake mechanism. Every single connection (each subflow of MPTCP) employs a different IP address in end-to-end connection [15]. In end-hosts, MPTCP is set in the transport layer that is hidden from both the network layer and application layer, as shown in Fig. 2. A logical overview of MPTCP in end-host is presented in Fig. 3. Accordingly, we can divide the MPTCP structure into three main sections, including path manager, path/packet scheduler, and congestion control [16].

Path manager: The path manager detects and uses multiple available paths between two end hosts based on defined criteria. The MPTCP employs multiple IP addresses as an indicator of end hosts at one or both sides of the connection. The sender node starts MPTCP connection by establishing a single-path TCP connection. Then, the multi-homed node adds more subflows with “ADD ADDR” command.

Congestion control: The congestion control method guarantees that an MPTCP algorithm cannot unfairly use network bandwidth compared to a single-path TCP stream in a shared media. Some commonly used congestion control protocols for MPTCP are OLIA, BALIA, and D-LIA, which employ Additive Increase and Multiplicative Decrease (AIMD) method.

Path scheduler: The path scheduler is one of the most significant functions of MPTCP that affects the performance of MPTCP, especially in challenging applications such as M2M systems and IoT networks. An MPTCP scheduler is applied to distribute packets of user data between all available subflows. Path scheduler also has a significant effect on the network performance, in particular when subflows are heterogeneous [8,12]. MPTCP scheduler in the sender node starts when the data stream arrives at the transport layer from the application layer. The decision of which packets must be conducted on which subflow is based on different metrics. Most schedulers decide according to metrics delivered by the network layer, such as CWND and RTT. The remaining steps of the common scheduling procedure are presented in Fig. 4.

In this paper, we focus on a path scheduler to design a proper scheduling algorithm for MPTCP in wireless networks. The OfO packet is one important performance-based issue in the MPTCP scheduler due to receive window restrictions. When the receiver node does not have

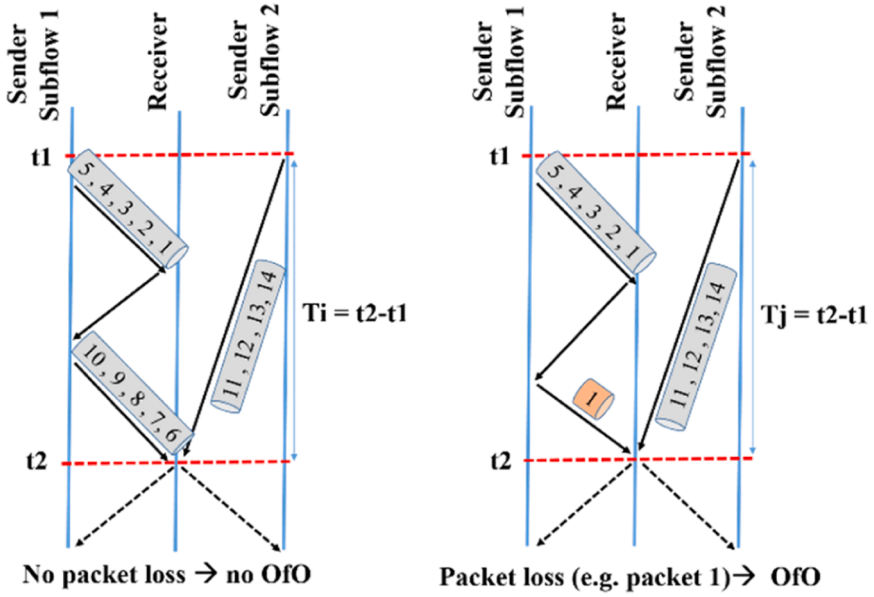


Fig. 5. Packet loss and OfO in MPTCP.

sufficient buffer to save all OfO packets, retransmissions occur because of request time out, and the loss ratio increases drastically. The main reason for the OfO issue is that each involved subflow has a different packet loss, network bandwidth, and subflow delay [6]. For instance, we assume that we have 15 packets to send at packet scheduling time T (where $T = t2\text{-}t1$) as shown in Fig. 5. The RTT of subflow1 and subflow2 can be 10 ms and 20 ms, respectively, and CWND of both paths is 5. If a packet (packet number 1) is missing in the subflow1, the sender node must retransmit the lost packet in the following transmission according to the recovery procedure. Therefore, packets 6–10 (from subflow1) cannot reach the receiver node before packets 11–15 (from subflow2) because the sender node must recover the missing packet in the next round. Accordingly, the packet loss increases the number of OfO packets in the system considerably [12,14]. Thus, transmitted packets can reach out-of-order to the receiver node and need to be reordered. To solve this issue (OfO packets) over heterogeneous paths, a proper scheduling plan should be used to reduce transmission delay, reduce communication costs, and increase network throughput [5].

3. Literature review

Many authors have studied the MPTCP and worked on different sections of MPTCP, in particular, the MPTCP scheduler. We present a short review of some of the related works in the MPTCP. Li et al. [26] have evaluated the functionality of MPTCP in different wireless systems like cellular networks by means of a real testbed. The authors indicated that MPTCP is much better than standard TCP, especially in some delay-sensitive applications. Habib et al. [27] and Quentin et al. [28] have shown the main differences between single-path protocol and MPTCP in detail. The authors provided a list of multiple features of MPTCP, such as reduced delay, improved throughput, and load balancing.

Han et al. [29] have presented an overview of MPTCP in a real testbed environment. The authors listed the benefits of MPTCP in mobile systems and obtained fewer delays by selecting paths with lower RTT. Grinnemo et al. [30] have provided a survey on cloud-based MPTCP in mobile communication systems to show how the latency can be reduced by using MPTCP. By using a testbed, the authors indicated that MPTCP can improve the performance of cloud-based systems properly. Houze et al. [31] have shown that the quality of online video streaming is based on latency between video production and playback. The authors suggested the implementation of MPTCP-based video distribution at the application layer to decrease latency properly. Li et al. [32] and Polese et al. [33] have presented taxonomy, characteristics, issues, and solutions of MPTCP transmissions for various layers of the protocol stack.

Different CCAs have been suggested for MPTCP to solve the main performance issues in the system [34–38]. The suggested approach designed an arithmetic model to control the increase of path's CWND to balance the congestion level properly. In [39], authors have provided an overview of MPTCP-based video streaming in different wireless networks to show the advantages of MPTCP in wireless systems. In [40–44], the authors have presented a comparison between MPTCP and other transport layer protocols in the emulated and real-world environment for mobile systems. The authors in [45–47] have discussed the most significant aspects of transferring from single-path TCP to the MPTCP. They listed several merits of MPTCP and its main features, like load balancing, to estimate the expected performance of MPTCP in various environments. Based on these studies, the MPTCP provides better throughput compared with TCP. Yedugundla et al. [48] have evaluated and compared CMT-SCTP and MPTCP as transport layer protocols in different applications such as video streaming, web browsing, and gaming traffic.

In recent years, many authors have studied packet scheduling algorithms in the MPTCP. The main goal is the performance-based issues of the Default scheduler (minRTT) in the MPTCP. Accordingly, several scheduling solutions with different goals are presented to improve the MPTCP performance [49–64]. Some authors employ all available paths, while others use only one or several active subflows out of all available subflows in their suggested approaches. In some solutions, authors employed the buffer size of the sender as a selection factor, while some other authors employed the buffer of the receiver as a path quality estimation. The main idea of these proposed schedulers is to save more OfO packets. However, increasing buffer size can lead to increased delay, particularly in real-time applications [64]. The majority of presented approaches estimate path efficiency by using different characteristics such as RTT, CWND, and queue size. In addition, the MPTCP scheduler can be designed based on two metrics, including the buffer size of the router and the buffers of each end host [52]. Some proposed schedulers give the highest priority to the new packets over the resent user's packets for the duration of data transmission from the sender to the receiver.

Some path schedulers retransmit a set of new packets on a fresh route

Table 1
An evaluation of MPTCP schedulers.

Schedulers	Core idea	Advantages	Disadvantages
Round Robin	Subflows are employed alternately, one path after the other one.	It provides capacity aggregation and optimal load balancing. It works well in symmetric subflows.	It cannot work efficiently in heterogeneous paths (asymmetric subflows).
MinRTT	It uses the fastest subflow with the shortest RTT among all available paths and enough CWND space.	It is easy to implement in different MPTCP applications.	It cannot work properly with asymmetric paths. It is not able to estimate the number of packets for each subflow appropriately.
Redundant	It utilizes all available subflows to send the same packets.	It is speedy and has lower latency.	It creates much more overhead in the buffer.
DAP	It uses all subflows to reduce buffer blocking.	It is good for wireless paths.	It does not deal with network failures.
ECF	It improves the throughput by increasing the use of the fastest path. It avoids using a slow path even if the fast path is busy.	It has fast completion time that is good for heterogeneous networks.	It can increase the number of OfO packets in the receiver node.
OTIA	It assigns each segment to the path with the smallest departure time to minimize the required transmission time.	It needs fewer parameters and reduces transmission time, which works for real-time applications.	It suffers from OfO packets. It cannot schedule reinjections.
BLEST	It avoids sending packets on a slower path if the faster path can be available again.	It can work well for heterogeneous paths and decreases HoL blocking and OfO packets.	It has error sensitivity and suffers from significant performance degradation when losses occur frequently.
STTF	It assigns unsent packets to each path based on transfer time.	It works well for heterogeneous subflows.	It suffers from OfO packets.

in place of the previous subflow. A set of some commonly used MPTCP schedulers are compared in [63] by means of the Mininet emulation tool. The authors identified asymmetry as one of the main reasons for reduced performance in MPTCP schedulers. Wang et al. [12] have proposed the Slide Together Multipath Scheduler (STMS) method to decrease OfO packets and solve the problem of limited buffer size. However, the STMS does not focus on application performance properly. Shreedhar et al. [52] have suggested an MPTCP scheduler named QAware, which employs information on lower OSI layers such as link quality and utilization of each path. The main idea of QAware is that a given path which is used frequently increases its end-to-end delay, and it is less attractive to be selected as subflow. The QAware is evaluated and compared by using both testbed and simulation tools.

Another MPTCP scheduler for bulk data transmission has been proposed by Kimura et al. [55]. The authors have presented that the suggested procedure can increase MPTCP throughput by using sending rate and window space as effective metrics to select available subflows. Xue et al. [53] have defined another scheduler based on the expected amount of traffic for each available path. They checked the proposed scheduler by means of the NS3 simulation tool. The authors detailed evaluation results concentrating on the throughput performance of the system. Frommgen et al. [54] have discussed that the schedulers that are according to the RTT value suffer from stale RTT value, especially for light stream data like HTTP traffic. Silva et al. [50] have presented the scheduling method based on the variation of kinds of transmitted data that is used in Augmented/Virtual Reality technologies. The suggested method is evaluated by using the NS3 simulation tool. Nagayama et al. [49] have presented another MPTCP scheduler that includes information on TCP session state in the subflow selection algorithm. In the proposed method, the authors minimize subflow switching in the scheduling algorithm to make an efficient selection.

In addition, a set of most common schedulers is presented in this paper, including MinRTT, ECF, Redundant, Shortest Transmission Time First (STTF), RR, Delay-Aware Packet (DAP), Loss Aware, Out-of-order Transmission for In-order Arrival (OTIAS), and Blocking Estimation-based (BLEST). However, these methods did not consider retransmission issues, causing an increase in the OfO packets and a deacrea in the total throughput in wireless networks where an MPTCP connection has subflow heterogeneity. In Table 1, we provide a comprehensive overview of the related works of MPTCP schedulers. Many scheduling algorithms are proposed for the MPTCP, but we listed the most common schedulers with their core ideas, advantages, and disadvantages. In addition, Table 2 summarizes the schedulers, highlighting their applications, overall goals and metrics which make scheduling decisions.

MinRTT Scheduler: The default scheduler in the MPTCP, named MinRTT, works by means of the RTT parameter [8,60]. The MinRTT scheduler transmits packets through the available subflow which has lower RTT, and then sticks to the path until the CWND of this subflow becomes full of packets. Then, the scheduler employs a second subflow which has the next higher RTT. Accordingly, the number of transmitted packets on selected subflows is based on its CWND. The MinRTT algorithm cannot consider packet ordering appropriately. Suppose there are two paths named subflow1 and subflow2, as shown in Fig. 6. The RTT of the fast path (RTTf) and RTT of the slow path (RTTs) are 5 ms and 10 ms, respectively. The congestion window size of both paths is the same as each other (i.e., CWND=10). The required time for a slow subflow is double the required time for a faster subflow. When there are 25 packets to transfer, according to the MinRTT scheduler, packets 1–10 and 21–25 can be conducted on a faster subflow, and packets 11–20 can be sent on a slower subflow. The MPTCP receiver node (MPTCP server) needs to wait for packets of the slower path for a long duration.

Earliest Completion First (ECF) Scheduler: The ECF addressed the performance degradation issue of the subflow asymmetric by minimizing the waiting time of the fast subflow [2,65]. The ECF scheduler considers the RTT of each subflow, the size of the transmitted packet, and the available CWNDs. Accordingly, the ECF selects the fastest path with the shortest RTT on the condition that it can send on that path. Then, it finds the next fastest subflow with enough CWND. To take the decision for selecting the available path, ECF checks the arrival time for a packet on both paths. When the arrival time is at least double the RTT of the fastest path, ECF decides to ignore the slow path and wait for the fast path instead. However, the ECF still is not able to provide in-order delivery of the packets to the receiver node leading to OfO packet issue and decrease the total throughput. The details of the ECF scheduler are given in [13].

Redundant Scheduler: This scheduler can provide proper redundancy in the MPTCP for conducting user's data over available paths.

Redundant Scheduler: This scheduler can provide proper redundancy in the MPTCP for conducting user's data over available paths. The redundant scheduler is implemented in the MPTCP Linux Kernel implementation and sends all packets on every available path with enough space in CWND as shown in Fig. 7. Then, the receiver nodes send back a combined acknowledgement of received segments on each path.

However, transferring the same packets over all available subflows can waste network Bandwidth (BW) drastically. Thus, the network overhead can be the main challenge for redundant schedulers, especially in mobile communication systems which has not unlimited resources [2,

Table 2
Comparison of scheduling algorithm in MPTCP.

Schedulers	Goals	Application	Metrics						
			CWND	RTT	MSS	Loss	Subflow ID	Inflight packets	Data segment size
ESPC [66]	Improve throughput	Bulk transfer	✓	✓		✓			
LL [1]	Improve throughput	Bulk transfer		✓					
LTS [55]	Improve throughput	Bulk transfer	✓	✓				✓	
LWS [55]	Improve throughput	Bulk transfer	✓	✓				✓	
HSR [55]	Improve throughput	Bulk transfer	✓	✓	✓				
WRR [59]	Optimize load balance	Bulk transfer	✓						✓
BLEST [10]	Minimize retransmissions, mitigate HoL-blocking	Bulk transfer	✓	✓	✓			✓	
ECF [13]	Maximize fast subflow utilization	Streaming video transfer	✓	✓					
DMPTCP [67]	Reduce completion time	Short flow transfer	✓	✓					✓
STTF[60]	Reduce completion time	Web and interactive app.	✓	✓					✓
OTIAS [18]	Reduce completion time, mitigate jitter	Real-time transfer	✓	✓				✓	
LL/RP [63]	Mitigate HoL-blocking	General applications	✓	✓					
DAPS-2 [19]	Mitigate HoL-blocking	General applications	✓	✓				✓	
RR [1]	Academic/testing goal	General applications					✓		
LL/BM [63]	Mitigate bufferbloat	General applications	✓	✓					

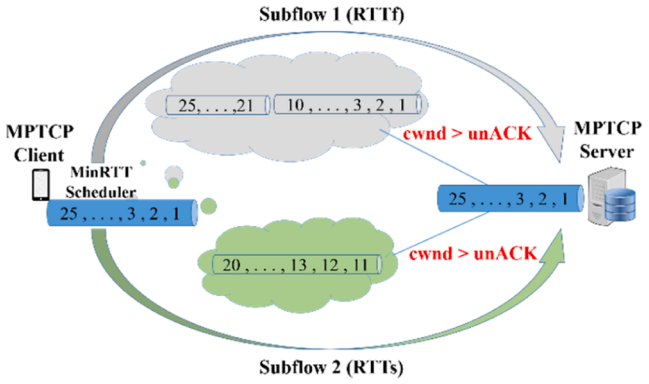


Fig. 6. MinRTT scheduler.

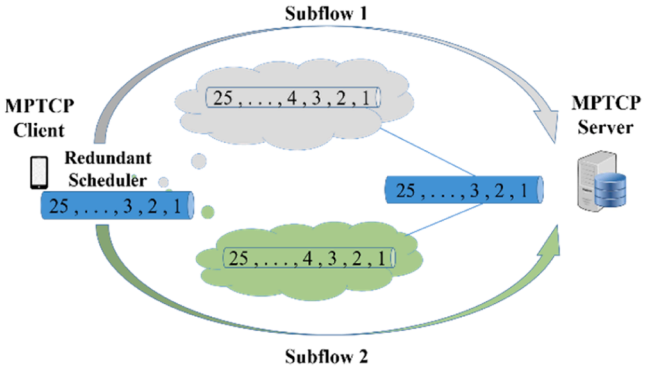


Fig. 7. Redundant scheduler.

65].

Shortest Transmission Time First (STTF) Scheduler: The main idea of the STTF path scheduler is that packets will be assigned to the available subflow, which needs the shortest estimated time to send the packet to the receiver node. By using the estimated time, the STTF scheduler employs the current congestion control state to check both CWND and RTT of each path. Accordingly, the STTF scheduler schedules each unsent packet through the fastest active subflow, even if its CWND has no space. The main hint is that when the packet is ready to send, it

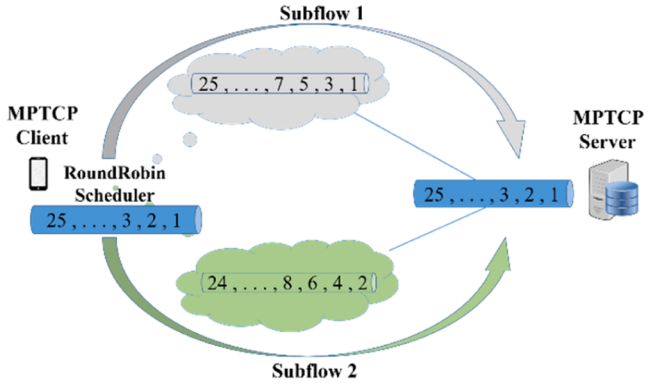


Fig. 8. Round-Robin scheduler.

sends all unscheduled packets. Then, it estimates the transfer time for each of the paths and finds on which paths each packet must be scheduled.

This estimation is according to the CWND of each path, the state of the path (i.e. congestion avoidance), and the number of queued segments in each path [65].

Round-Robin (RR) Scheduler: RR scheduler will not prefer any available path to send the packets. It employs subflows one after another without preferring (the smallest RTT or BW). RR scheduler begins by

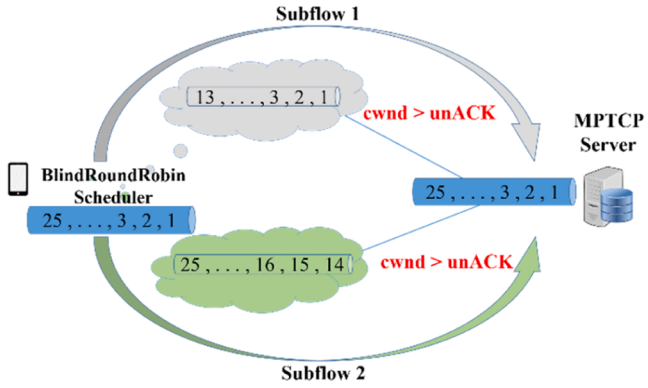


Fig. 9. Blind Round-Robin scheduler.

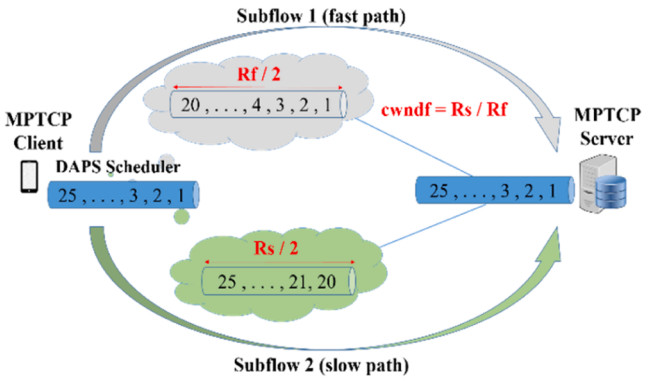


Fig. 10. DAP scheduler.

filling the CWND of each path one by one in turn. Fig. 8 shows the RR scheduler. We assume that there are two paths. The first path (subflow 1) sends packets 1, 3, 5, …, 25, while the second path (subflow 2) forwards packets 2, 4, 6, …, 24 [7,11]. However, when we have bulk transfer, this path scheduler cannot detect a true RR packet assignment. In addition, when there are heterogeneity in subflows, the performance of the RR scheduler is decreased drastically because each slow path can be limited by its CWND.

Thus, this scheduler is not able to perform properly in heterogeneous networks. Blind RR is a common version of this path scheduler. It checks three parameters, including receiver CWND, inflight packets, and unacknowledged packets. When receiver CWND is filled in each path, then all paths are blocked for a given period. Blind RR is shown in Fig. 9, and more details are presented in [65].

Delay-Aware Packet (DAP) Scheduler: The main idea of the DAP scheduler is to determine available subflows based on CWND and forward the delay of each path. According to the difference between subflows, the DAP scheduler determines the number of packets for the next round of each subflow. In addition, this path scheduler can specify which packets must be sent through which paths by using forward delay and CWND [18,65]. Accordingly, the DAP scheduler replaces the RTT with the forward delay, which is the sum of sending time and in-flight time. Likewise, this scheduling algorithm has three phases; in phase 1, the scheduler determines the order of available subflows for transmission of packets; in phase 2, the scheduler determines which packets must be sent over each subflow; in phase 3, each packet can be sent based on the determined schedule in previous phases. DAP scheduler is shown in Fig. 10. The details of this path scheduler are described in [19].

The main objective of DAP scheduler is to transmit packets from sender to receiver in order. However, it does not consider path loss in MPTCP properly. The DAP scheduler does not exhaust fast paths completely, but it assigns only a given number of packets to fast paths and other packets to slow paths. Additionally, this scheduler works well when we have a stable RTT and CWND for the whole duration of transmission and cannot deal with unstable networks.

Loss Aware Scheduler: The main idea of this path scheduler is to perform switching between MinRTT and Redundant schedulers according to path loss of subflows. It can develop the MinRTT by checking packet loss. Accordingly, it improves MPTCP performance, especially in reducing bandwidth overhead. The shortest transfer time of each path must be used for packet transmission, and the scheduler selected is MinRTT. When the path loss factor of each path increases and reaches a defined threshold, the Loss Aware scheduler switches to the Redundant scheduler [9,65].

Out-of-order Transmission for In-order Arrival (OTIA) Scheduler: The idea behind the OTIA scheduler is that the packets must be scheduled on the available path with the shortest transfer time. When a fresh segment is conducted, the OTIA scheduler estimates the arrival time of that packet at the receiver side to order subflows based on the shortest arrival time. The OTIA scheduler assumes that the one-way delay time can be RTT/2. This approach shows that it can schedule the packets on all available paths even if the subflows do not have enough CWND space. To determine the transmission time of the paths which have no available CWND space, the OTIA scheduler computes the number of RTTs for the queued packets on those paths. In each path, these queues can be constructed according to the assumption that these queued packets will be transmitted when the CWND of each path is available. However, the OTIA scheduler does not consider packet reinjection when a given subflow blocks the connection for any reason [3,8,18].

Blocking Estimation-based (BLEST) Scheduler: The most significant idea of the BLEST scheduler is that the scheduler can control the buffer blocking and avoid unwanted retransmissions [10]. The BLEST scheduler dynamically estimates the amount of data to send over each available path and makes the packet transmit through the fastest path, even if the slow path has enough CWND space to avoid buffer blocking in the receiver node. Likewise, it can improve application performance in MPTCP scenarios.

The BLEST scheduler algorithm assumes that each packet in flight on the slow path occupies space of CWND for the same amount of time, and these packets are acknowledged previously. The faster path, which has a lower RTT, utilizes the remaining MPTCP's Send Window (MPTCPSW). Likewise, the HoL blocking can occur where a faster path is not able to send because of limited MPTCPSW. The scheduler checks the amount of packets named X that can be conducted over the fast path in RTTs and checks if X can fit into MPTCPSW.

The BLEST scheduler starts with the MinRTT scheduler. Then, when the CWND fills, it checks the slow path to compute X to send over a slower subflow, and then it evaluates that it should wait for free CWND in the next round of the fast path or send the existing packets over the slow path. This evaluation is based on the CWND of fast subflow (CWNDf), RTTs/RTTf (i.e., rrts), Maximum Segment Size of fast path (MSSf), inflight data over slow path, and MPTCPSW. It can monitor the MPTCPSW and decrease the required time when the fast path is not able to be used because of insufficient CWND space. The main objective of the BLEST scheduler is to improve MPTCP's performance by reducing OfO, especially in heterogeneous subflows. However, it cannot consider idle fast paths, which can reduce the utilization of available fast subflow drastically.

4. Proposed scheduler

To solve the main challenges of MPTCP, as mentioned earlier, we focus on the MPTCP scheduler and design a practical and robust scheduler to improve the key performance metric of the network appropriately. This section presents the details of our proposed scheduling algorithm, PRS-MPTCP. The main idea behind the proposed scheduler is to have efficiency improvement of MPTCP, especially when there is path dissimilarity between the MPTCP subflows. The proposed algorithm addresses the challenges of decreasing the number of OfO packets and increasing network performance in heterogeneous scenarios. The proposed scheduler inherits the benefits of the existing default scheduler and improves the drawbacks by considering new metrics in the meanwhile.

The suggested PRS-MPTCP scheduler improves the performance of MPTCP by the number of retransmitted packets, retransmission rate, and the number of inflight packets of each path as new metrics in the scheduling policy alongside the usual metrics used by other existing algorithms such as RTT and CWND. The retransmission rate shows the proportion of packets that need to be retransmitted, which can significantly impact the throughput of a network. By considering the retransmission rate of different subflows and dynamically optimizing their scheduling decisions, the proposed scheduler can enhance the MPTCP connection's performance, resulting in higher throughput, a smaller number of OfO packets, and better efficiency, especially in

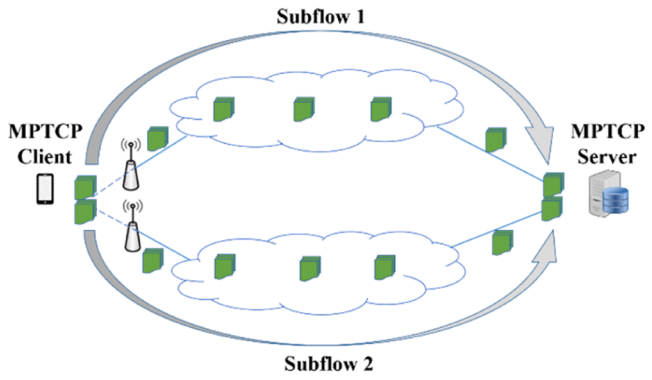


Fig. 11. The subflows of the PRS-MPTCP.

heterogeneous networks. Additionally, it uses the number of inflight packets and the size of the CWND of each subflow to check the possibility of sending data packets and find the free space in the congestion window of subflows. Comprehensive explanations of the proposed algorithm have been included in the rest of this section to describe the decision-making criteria of the PRS-MPTCP scheduler.

The algorithm comprises two sections: 1) checking the number of available subflows, and 2) checking different states for each subflow to find the best path according to the current network condition. Given particular path characteristics, including RTT, CWND, inflight packets, and retransmission rate, the new model finds the best subflow to send data packets and increases the total performance. In the PRS-MPTCP scheduler, we assume that there are packets in the MPTCP connection-level send window which are waiting to be sent over the best subflow.

The presented algorithm is formulated for the most common MPTCP scenarios with two different categories of subflows in terms of RTT (RTT of fast and slow subflow) [3,5,53]. As can be seen in Fig. 11, the MPTCP client distributes user data to the available subflows based on a defined scheduling policy. On the receiver side, the MPTCP server reassembles received packets and sends the data to the application layer.

The procedure of the proposed PRS-MPTCP scheduler is presented in this section to show the main steps of the scheduling algorithm. Fig. 12 indicates how our scheduler is designed according to the characteristics of each path. The used variables in the PRS-MPTCP algorithm are listed in Table 3. Fast subflow has lower RTT, and slow subflow has higher RTT. In the first step, the algorithm finds the fastest subflow, S_f. Then, the scheduler finds the best subflow using the default scheduler's policy. After that, the algorithm checks the similarity of these subflows. If they are the same, it means that only S_f is available. Therefore, PRS-MPTCP can select S_f as the best path and efficiently send the data segment over it.

In this regard, the PRS-MPTCP algorithm, before checking other conditions, first finds some specified metrics for each available subflow. We consider inflight packets of each subflows (i.e., *inflight_f*, and *inflight_s*), and CWND of each path (i.e., *cwnd_f*, and *cwnd_s*) in the first steps of PRS-MPTCP. The conditions are designed according to the free

Table 3
Variables used for PRS-MPTCP scheduler.

Variables	Corresponding definition of each variable in the model
S_s	The slow subflow with the largest RTT
S_f	The fast subflow with the smallest RTT
inflight_s	The number of inflight packets in slow subflow
inflight_f	The number of inflight packets in fast subflow
cwnd_s	The congestion window size of slow subflow
cwnd_f	The congestion window size of fast subflow
space_s	The remaining space of cwnd_s
space_f	The remaining space of cwnd_f
performance_s	The performance in terms of retransmissions of slow subflow
performance_f	The performance in terms of retransmissions of fast subflow
retrains_s	The number of retransmitted packets of slow subflow
retrains_f	The number of retransmitted packets of fast subflow
total_transmited_s	The total number of transmitted packets via slow subflow
total_transmited_f	The total number of transmitted packets via fast subflow
retrans_rate_s	The retransmission rate of slow subflow
retrans_rate_f	The retransmission rate of fast subflow
rtt_s	The round-trip time of slow subflow
rtt_f	The round-trip time of fast subflow
S_{best}	The best subflow in terms of throughput and OfO

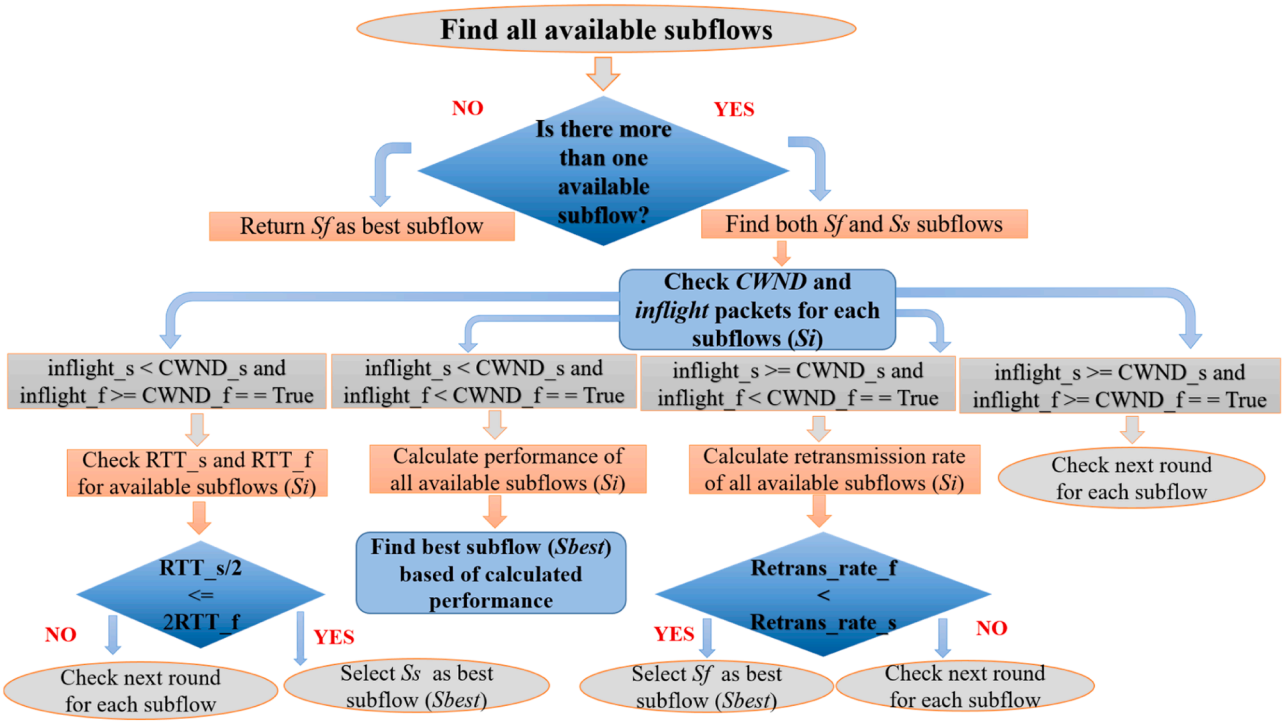


Fig. 12. Flowchart diagram of the PRS-MPTCP.

```
Algorithm: PRS-MPTCP scheduler
  Input  : Set of subflows, S
  Output: The best subflow, S_best ∈ S
1: if only the fastest subflow S_f (with smallest RTT) is available then
2:   S_best ⟵ S_f
3:   return S_best
4: else  both fast subflow S_f and slow subflow S_s are available in S
5:   foreach available S_i ∈ S  do
6:     find the current number of inflight packets and cwnd size of S_i
7:     inflight_i ⟵ tcp_packets_in_flight ( )
8:     cwnd_i ⟵ snd_cwnd
9:   end
10: if inflight_s >= cwnd_s and inflight_f >= cwnd_f then
11:   S_best ⟵ NULL
12:   return S_best
13: end
14: if inflight_s < cwnd_s and inflight_f < cwnd_f then
15:   space_f ⟵ cwnd_f - inflight_f
16:   performance_f ⟵ space_f rtt_f * max (1, retrans_f)
17:   space_s ⟵ cwnd_s - inflight_s
18:   performance_s ⟵ space_s rtt_s * max (1, retrans_s)
19:   if performance_f > performance_s then
20:     S_best ⟵ S_f
21:     return S_best
22:   end
23:   if performance_s > performance_f then
24:     S_best ⟵ S_s
25:     return S_best
26:   end
27: end
28: if inflight_s >= cwnd_s and inflight_f < cwnd_f then
29:   retrans_rate_f ⟵ retrans_ftotal_transmited_f
30:   retrans_rate_s ⟵ retrans_stotal_transmited_s
31:   if retrans_rate_f < retrans_rate_s then
32:     S_best ⟵ S_f
33:     return S_best
34:   else
35:     S_best ⟵ NULL
36:     return S_best
37:   end
38:
39: end
40: if inflight_s < cwnd_s and inflight_f >= cwnd_f then
41:   if (rtt_s2) <= (2*rtt_f) then
42:     S_best ⟵ S_s
43:     return S_best
44:   else
45:     S_best ⟵ NULL
46:     return S_best
47:   end
48: end
49: end
```

Fig. 13. The procedure of the PRS-MPTCP.

space in the congestion window of each subflow. The subtraction between *cwnd_i* and *inflight_i* (*cwnd_i - inflight_i*) can specify the free space in the network buffer of S_i at a specific time, where *i* refers to path number (i ∈ {1, 2, 3, . . ., *n*}, *n*=number of available subflows). Therefore, there is free space in the congestion window if *inflight_i* < *cwnd_i*. By keeping this comparison, we can define four different conditions for the MPTCP system. The first condition occurs when no free space exists in S_f and S_s. In this case, the algorithm does not return any of S_f and S_s as the best path, and it checks the conditions again. The second condition is when both S_f and S_s have free space in their CWND. In this case, the scheduling algorithm first calculates the free space of both subflows (S_f and S_s). Then, we use the value of free space to calculate a new metric named *performance_i*, where *i* refers to the path number. The performance metric (*performance_i*) can be calculated as (1):

$$performance\_i = \frac{space\_i}{rtt\_i * max(1, retrans\_i)} \quad (1)$$

where *space_i, rtt_i* and *retrans_i* demonstrate the remaining space of *cwnd_i* (S_i), round trip time, and the number of retransmitted packets in each subflow, respectively.

The main goal of this metric is to find out each subflow' s performance by measuring how effectively it makes use of the available network resources. A connection that performs better will make greater use of the capacity it has available, have a shorter round-trip length, and need fewer retransmissions. In addition, to ensure that the performance value is not negatively impacted in cases where there are no retransmissions, the "*max(1, retrans_i)*" portion of the formula is used. It takes the highest value, "*retrans_i*" between 1 and the real amount of retransmissions. Whenever there are no retransmissions, division by zero is omitted since it is undefinable. Accordingly, the algorithm selects the subflow with the higher performance as the best path and sends the user's data over it. The third condition occurs when only S_f has free space in its CWND. In this case, the algorithm does not send the data packet to S_f immediately. Instead, to make a more accurate decision, it calculates a new metric named retransmission rate (*retrans_rate_i*) by (2). The *retrans_rate_i* measures the percentage of packets that are retransmitted due to loss or congestion in the network, and it can be used to analyze the performance of a network.

$$retrans\_rate\_i = \frac{retrans\_i}{total\_transmited\_i} \quad (2)$$

By calculating the *retrans_rate_i* of both S_f and S_s, the PRS-MPTCP algorithm can gain insights into the quality of each subflow. A higher *retrans_rate_i* can illustrate that a considerable number of packets have been lost or corrupted during transmission. Therefore, if the retransmission rate of S_f is less than the retransmission rate of S_s (*retrans_rate_f* < *retrans_rate_s*), the scheduler sets S_f as the best subflow in the system. Otherwise, even if there is free space in *cwnd_f*, it does not send the data packet over S_f. Instead, the algorithm checks the defined conditions again. The fourth condition occurs when only S_s have free space in their congestion window. In this case, the scheduling algorithm extracts the RTT of each subflow as the corresponding performance metric. As can be seen in Fig. 13, *rtt_f* and *rtt_s* indicate the round trip time of S_f and S_s, respectively. Accordingly, if *(rtt_s)/2* ≤ *2*rtt_f* is satisfied, S_s can be selected as the best subflow in the system.

By taking the conditions mentioned above into account, the proposed algorithm finds the best path dynamically for each segment according to the present characteristics of each path and the network conditions. Therefore, after selecting the best path for each segment, the PRS-MPTCP scheduler ensures the optimization of the performance of MPTCP in heterogeneous scenarios by assigning each segment to the path with the best performance.

A detailed flowchart diagram illustrating the PRS-MPTCP scheduler's structure is shown in Fig. 12. In Fig. 13, we present the corresponding details. We include additional information on the proposed algorithm in Fig. 13 along with a pseudocode representation. In the presented pseudocode, we specified the Fast and the Slow Subflows as S_f and S_s, respectively.

The proposed PRS-MPTCP scheduler can improve MPTCP performance by mitigating the effects of retransmission in wireless networks. Retransmission is a major issue in wireless networks that can negatively affect MPTCP performance. This results from packet loss or corruption, which is a prevalent problem in wireless networks and wastes the resources of the network. As a result, it qualifies as a significant

contributor to throughput degradation. The delivery of next packets will be delayed, and retransmission will be required in the event of a packet loss in one subflow. As a result, that particular subflow's throughput will drop. The entire MPTCP performance is affected by this throughput reduction on a single path. The number of retransmission packets and the retransmission rate of each path are taken into account by the proposed PRS-MPTCP scheduler when making scheduling decisions. With the help of this dynamic modification, the proposed scheduler can be guaranteed to use pathways efficiently and reduce the impact of retransmissions on throughput overall. We ran several tests to evaluate the performance of the suggested PRS-MPTCP scheduler. The specifics of the experimental design and the evaluation's findings are covered in the following sections.

5. Experimental setup details

In this section, we outline various experiments to evaluate the proposed scheduler's performance. To conduct these experiments and facilitate comparisons with other schedulers under different scenarios, we modify the Linux Kernel and implement the proposed scheduler. We use version 0.96 of the MPTCP Linux Kernel implementation as the common base for all our implementations. This specific version (v0.96 release) is compatible with the Linux Kernel Longterm Support release v5.4.230. In Fig. 11, we present a standardized MPTCP architecture with a client and server, each equipped with two interfaces. We utilize Mininet-Wifi to conduct different experiments with different schedulers and evaluate the performance of MPTCP. For wireless network emulations, the Mininet-WiFi emulator offers an extensive feature set and a multitude of customization choices.

To see whether PRS-MPTCP is able to reduce the impact of subflow heterogeneity on performance, we must adjust different loss rates and RTT for different subflows. To achieve varying loss rates and RTT for each subflow, the Mininet-WiFi emulation environment enables us to set up a virtual wireless network on a single computer and use virtual hosts for the client and server running MPTCP v0.96. We utilize a Python script to generate the network topology and manage the virtual wireless network. We can define hosts, switches, and access points using the Python script and then connect them in the topology. In the script, the end nodes (sender and receiver) are connected via links with adjustable data rates and delays in a point-to-point communication design. They are connected via two wireless interfaces (independent MPTCP subflows) in the access network, with a configured link bandwidth of 100 Mbps and RTT ranging from 10 ms to 100 ms.

Our experimental evaluations are based on three distinct scenarios, where we observe changes in RTT and loss rate for the involved subflows. In the first scenario, we consider an RTT difference between subflows with a fixed loss rate during the transmission. In this scenario, we set variable RTT ratios (RTT_s to RTT_f) at 1, 4, 8, and 10, where the subflows experience a constant loss rate of 1 %. The second scenario takes into account how various loss rates affect transmission performance when both subflows have the same value of RTT. To evaluate the performance in this instance, different loss rates (LR = {0 %, 0.5 %, 1 %, 3 %, 5 %}) are set for subflows when the RTT of both subflows is the same (RTT = 10 ms). The RTT and loss rate values for the two subflows in the third scenario differ. In this scenario, the subflows have different RTT values, with the constant ratio of RTT_s to RTT_f at 4 (where $RTT\_s = 40$ ms and $RTT\_f = 10$ ms). Three sets of values for the loss rates are set for the subflows. In the first case, the slow and fast subflows have a 0.5 % and 1 % loss rate, respectively. In the second case, the slow and fast subflows have a 1 % and 3 % loss rate, respectively. In the third case, the loss rate of the fast subflow is 5 %, while for the slow subflow, the loss rate is set at 3 %.

To analyze the behavior of each scheduler algorithm, a 10 MB file is transmitted from the MPTCP sender to the MPTCP receiver throughout each experiment. This 10 MB file is sent using the iperf tool, which creates a TCP data stream because MPTCP treats all subflows like regular

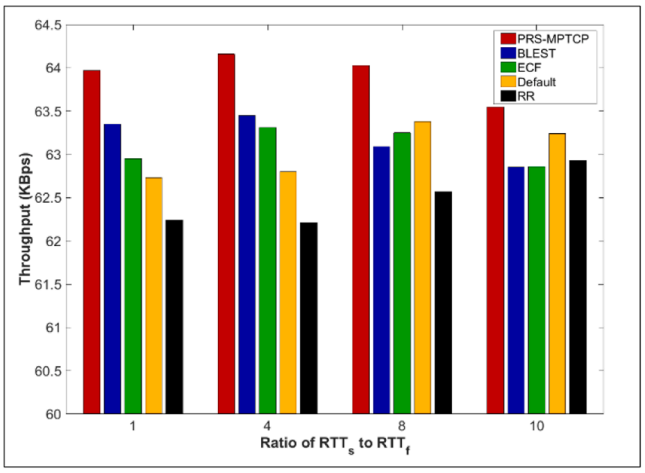


Fig. 14. The throughput of different schedulers.

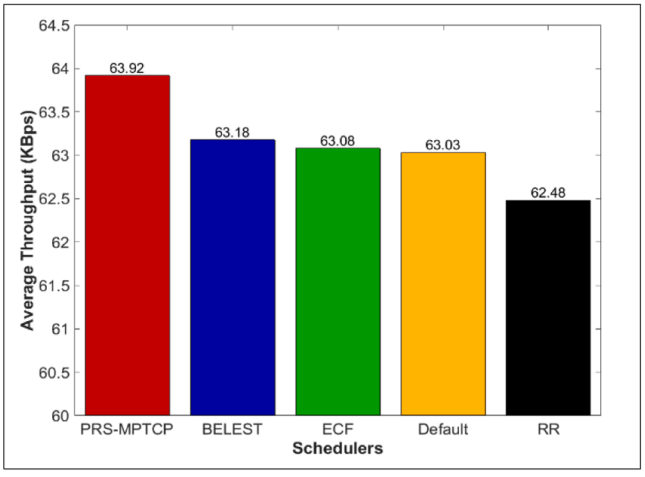


Fig. 15. The average throughput of different schedulers (LR = 1.00 %).

TCP connections. Additionally, this tool can function as a client and server and transfer data streams between them. A network protocol analyzer can help us better understand network performance by using statistics and analytical capabilities.

We used an open-source and free tool named Wireshark 3.2.3 as a network packet analyzer to record and analyze network traffic. We run this analyzer on the server side to capture the traffic in real-time and evaluate performance metrics such as throughput, the number of OfO packets, and the retransmission rate in different network scenarios. To obtain a more accurate evaluation, we run the scenarios for each scheduler ten times and calculate the average of the results [5,24,53].

6. Results and discussion

In this section, We conducted a comparison between PRS-MPTCP and four state-of-the-art MPTCP schedulers, namely ECF [13], BLEST [10], RR [1], and the default MinRTT scheduler [1]. To assess the performance of these algorithms in terms of throughput, the number of OfO packets, and retransmission rate. Our experiments were based on two key network parameters: loss rate and the difference of RTTs. The evaluation results showed that the proposed scheduler (PRS-MPTCP) outperformed the other algorithms in various scenarios.

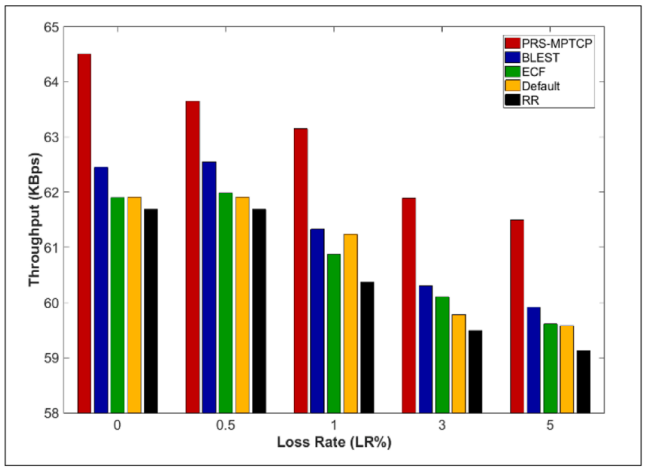


Fig. 16. The throughput of different schedulers based on LR.

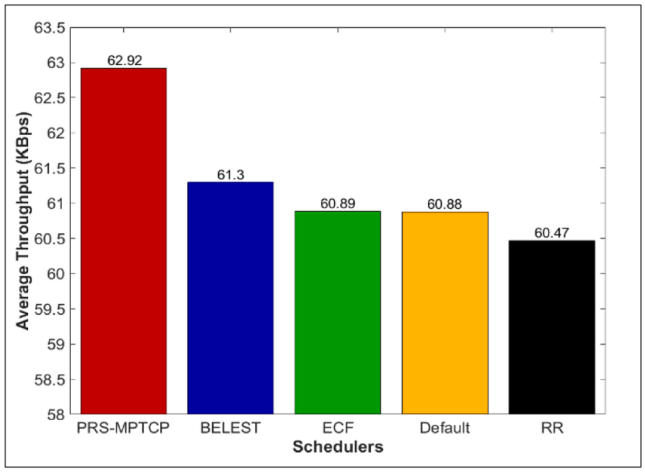


Fig. 17. The average throughput of different schedulers.

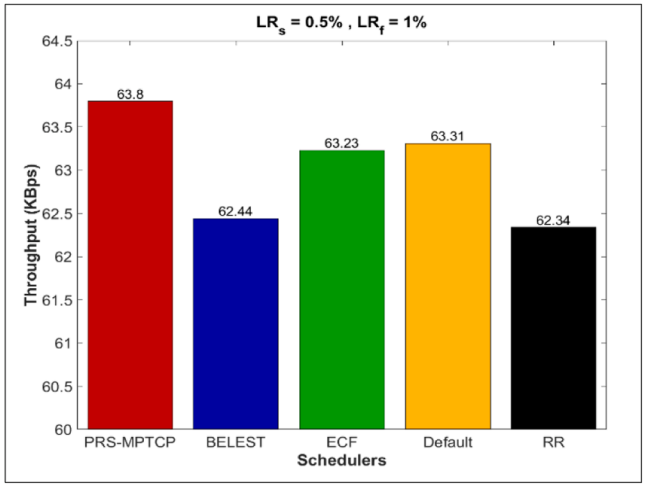


Fig. 18. The throughput of different schedulers based on different LRs (LR_s = 0.5 %, LR_f = 1 %).

6.1. Throughput of the schedulers

The throughput of an MPTCP connection between sender and receiver nodes was evaluated using various schedulers, as depicted in Fig. 14–18. As previously mentioned, we defined three scenarios based on different RTT values and loss rates. In the first scenario, we assessed the throughput (in KB per second) of each scheduler with varying RTT ratios (RTT_s to RTT_f) set at 1, 4, 8, and 10. The comparative analysis of the different schedulers can be found in Table 4 and Fig. 14.

Based on the evaluation results, PRS-MPTCP performs better than the other four widely deployed schedulers, especially when the RTT ratio between subflows increases. The comparative results of average throughputs for each scheduler are depicted in Fig. 15, considering a Loss Rate (LR) of 1.00 %.

In the second scenario, we evaluate the throughputs of different schedulers under various LR values (0 %, 0.5 %, 1 %, 3 %, 5 %). The comparative evaluation of these schedulers is presented in Table 5 and Fig. 16, with both subflows having the same RTT of 10 ms. Fig. 16 illustrates the behavior of each scheduler with different loss rates during the transmission of 10 MB files.

As expected, we observe that the average throughput decreases as the loss rate increases. This is because a higher packet loss rate results in a more dynamic network. Notably, PRS-MPTCP outperforms the other schedulers, demonstrating higher throughputs in the presence of packet loss. Furthermore, we offer a comparison of the average throughputs for

Table 4
The throughput comparision of different schedulers.

Scheduler	Ratio of RTT_s to RTT_f			
	1	4	8	10
Default scheduler	62.73	62.80	63.38	63.24
BLEST	63.35	63.45	63.09	62.85
ECF	62.95	63.31	63.25	62.86
RR	62.24	62.21	62.57	62.93
PRS-MPTCP (Proposed)	63.96	64.16	64.03	63.54

Table 5
The throughput of different schedulers based on LR.

Loss Rate	Throughput of Schedulers (KBps)				
	Default scheduler	BLEST	ECF	RR	PRS-MPTCP (Proposed)
0.00 %	61.91	62.45	61.90	61.69	64.49
0.50 %	61.91	62.55	61.99	61.69	63.65
1.00 %	61.24	61.33	60.87	60.37	63.14
3.00 %	59.78	60.30	60.10	59.49	61.88
5.00 %	59.58	59.92	59.61	59.13	61.49

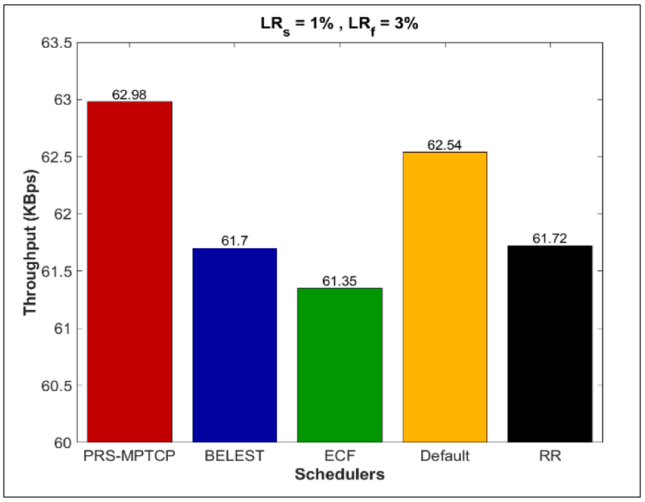


Fig. 19. The throughput of different schedulers based on different LRs (LR_s = 1 %, LR_f = 3 %).

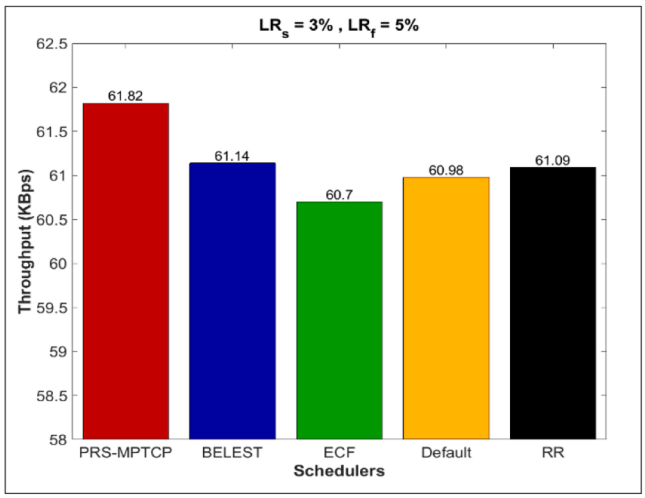


Fig. 20. The throughput of different schedulers based on different LRs (LR_s = 3 %, LR_f = 5 %).

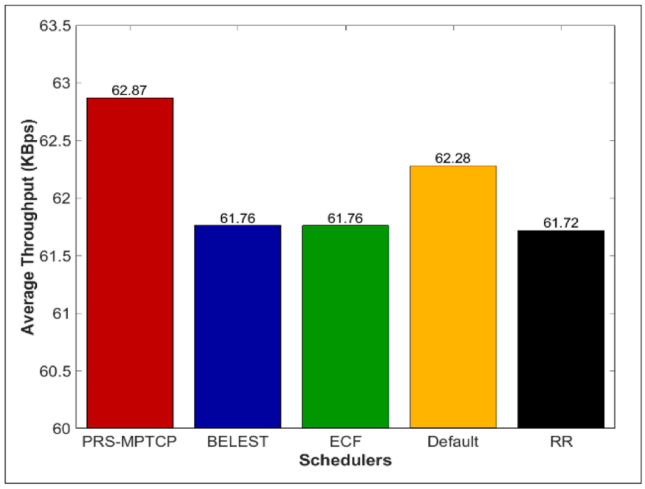


Fig. 21. The average throughput of different schedulers.

the aforementioned schedulers. As depicted in Fig. 17, the PRS-MPTCP exhibits a higher average throughput than the other schedulers.

Furthermore, in the third scenario, we assess the throughputs of the schedulers based on different LR and RTT ratios. We establish three distinct conditions for this experiment and evaluate each scheduler accordingly. In all conditions, the ratio of RTT_s to RTT_f remains constant at 4, while the LR of subflows varies. Fig. 18 illustrates the throughputs of different schedulers when LR_s (LR of slow subflows) and LR_f (LR of fast subflows) are set at 0.5 % and 1 %, respectively. Fig. 19 displays the throughputs of different schedulers when LR_s and LR_f are 1 % and 3 %, respectively. Moreover, Fig. 20 presents the throughputs of the schedulers when LR_s and LR_f are 3 % and 5 %, respectively.

As evident from the results, PRS-MPTCP significantly outperforms the other scheduler, consistently achieving higher throughputs compared to the alternatives. Additionally, Fig. 21 provides a comparison of average throughputs for these three conditions, clearly indicating a considerable increase in average throughput for PRS-MPTCP compared to the other schedulers.

6.2. OfO packets of the schedulers

In this section, we examine the number of OfO packets in different schedulers during the transmission of 10 MB files. The evaluation of the

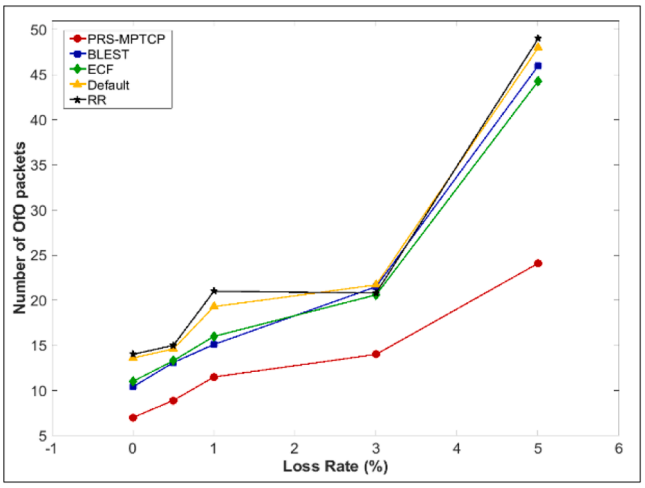


Fig. 22. The comparison of OfO packets in different schedulers (RTT = 10 ms).

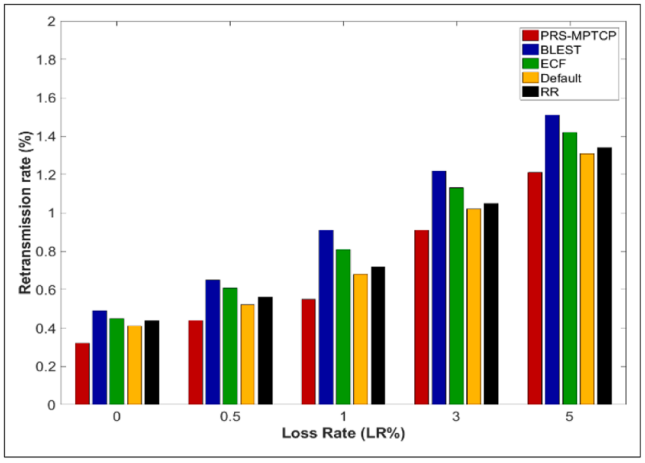


Fig. 23. The comparison of retransmission rate in different schedulers.

number of OfO packets is conducted under various LR scenarios (0.00 %, 0.50 %, 1.00 %, 3.00 %, 5.00 %). Fig. 22 presents the comparative analysis of different schedulers, with both subflows having the same RTT of 10 ms.

The comparative results demonstrate the superiority of our scheduling algorithm in reducing OfO packets in various asymmetric networks. As expected, the number of OfO packets increases with higher LR values. However, the results reveal that as the LR increases, the number of OfO packets in PRS-MPTCP remains significantly lower than in other schedulers, affirming the effectiveness of our proposed scheduler. In Fig. 22, it is evident that the proposed algorithm significantly outperforms the other schedulers, displaying the lowest number of OfO packets and the smallest slope. The superiority of our scheduler becomes even more pronounced as the LR increases. The comparative results demonstrate that PRS-MPTCP achieves average improvements of 38 %, 37 %, 45 %, and 44 % over BLEST, ECF, RR, and the Default scheduler, respectively.

6.3. Retransmission rate of the schedulers

Providing a comparative analysis of the retransmission rate is crucial for assessing performance in wireless networks. We evaluated the retransmission rate of PRS-MPTCP and compared it with other schedulers, including ECF, BLEST, RR, and the Default scheduler. The retransmission rate for each scheduler was assessed based on the second

Table 6
An overview of performance evaluation.

Scheduler	Performance metrics (Average)				
	OfO packets (count)	Retransmission rate (%)	Throughput (first scenario) (KBps)	Throughput (second scenario) (KBps)	Throughput (third scenario) (KBps)
Default scheduler	23.38	0.78	63.03	60.88	62.28
BLEST	21.12	0.95	63.18	61.30	61.76
ECF	20.86	0.88	63.08	60.89	61.76
RR	23.82	0.82	62.48	60.47	61.72
PRS-MPTCP	13.1	0.68	63.92	62.92	62.87

scenario, where both subflows have the same RTT of 10 ms under different LR values (0.00 %, 0.50 %, 1.00 %, 3.00 %, 5.00 %).

In general, as path diversity is available in MPTCP's scenarios, it can significantly decrease the possibility of a retransmission issue by sending the traffic to the other path. Additionally, the sending rate of MPTCP will be adjusted dynamically according to the network conditions using the congestion control algorithm. Therefore, considering these factors and the flexibility of MPTCP can help guarantee a reduction in the rate of retransmission of less than 1.5 % as illustrated in Fig. 23

The evaluation results reveal that our scheduler's retransmission rate (in percentage) is significantly lower than other methods. Fig. 23 demonstrates that as the loss rate increases, the retransmission rate for each scheduler also increases. However, the proposed method outperforms the other schedulers, as it exhibits the lowest retransmission rate compared to them in the presence of packet loss. The evaluation results show that the average improvements of PRS-MPTCP over BLEST, ECF, RR, and the Default scheduler are 28 %, 22 %, 16 %, and 12 %, respectively.

Based on the outcomes of our experiments, we find that the overall performance of the MPTCP is greatly improved by the proposed scheduling technique. In Table 6, an overview of our performance evaluation is presented.

7. Conclusion

Today, multi-homed devices come equipped with multiple interfaces, enabling them to utilize MPTCP for data transmissions. MPTCP offers an effective solution to enhance data delivery and improve resistance to wireless network failures in wireless networks. By employing multiple active interfaces for data transmission, MPTCP enhances the performance of network-based applications. This allows a sender node to extend its reach to a receiver node through multiple subflows, aiming to upgrade both network reliability and data transmission rates compared to conventional single-path TCP. However, MPTCP still faces challenges in achieving optimal performance, particularly in wireless systems where packet losses are prevalent, leading to rapidly changing path conditions.

A crucial factor affecting MPTCP's performance is the MPTCP scheduler, which aims to distribute unsent packets across available subflows. Incorrect scheduling decisions can increase OfO packets, especially when dealing with heterogeneous subflows. In this paper, we focus significantly on the path scheduler in MPTCP. Initially, we provide an in-depth overview of the state-of-the-art schedulers, including Default, DAP, ECF, OTIA, and BLEST, identifying and analyzing their main limitations.

Subsequently, we propose a robust scheduler to address the shortcomings of existing path schedulers and improve the performance of MPTCP subflows. We conducted a comparative evaluation of our proposed scheduler, PRS-MPTCP, through experimental analysis using our Linux implementation. We compare the performance of PRS-MPTCP with state-of-the-art path schedulers (MinRTT, BLEST, ECF, and RR) in terms of system throughput, retransmission rate, and the number of OfO packets.

Based on our experimental results, we observe that the proposed scheduling algorithm significantly enhances MPTCP subflow performance. PRS-MPTCP improves throughput in various scenarios with different loss rates, reduces the number of OfO packets, and lowers the retransmission rate considerably. Our evaluation demonstrates that our scheduler outperforms BLEST, ECF, RR, and the Default scheduler, achieving average improvements of 38 %, 37 %, 45 %, and 44 %, respectively. Moreover, our proposed scheduler exhibits the lowest retransmission packet rate, making better use of the available bandwidth, especially in lossy paths like wireless subflows.

The proposed PRS-MPTCP scheduler's design is considered a MPTCP connection with two active subflows which is the most common MPTCP scenario similar to the studies in [3,5,53]. In future work, we plan to extend the proposed algorithm to handle multiple (i.e., 3 or more) wireless interfaces in different network scenarios. To accomplish this goal, we might modify our scheduler lightly and employ some parameters differently. In this regard, we might need to modify scheduling policies to handle the increased complexity arising from a larger number of subflows.

CRediT authorship contribution statement

Atefeh Ahmadniai Khajekini: Writing – review & editing, Writing – original draft, Software, Methodology. **Hasan Amca:** Writing – review & editing, Supervision. **Ali Hakan Ulusoy:** Writing – review & editing, Supervision. **Enver Ever:** Writing – review & editing, Supervision.

Declaration of competing interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

Data availability

Data will be made available on request.

References

[1] M. Imtiaz, L. Tabassum, C. You-Ze, Performance evaluation of MPTCP on simultaneous use of 5G and 4G networks, Int. J. Sens. 22 (2022) 1–18.
[2] K. Rao, B. Jagadesh, Design and implementation of dynamic packet scheduling with waiting time aware: DPSW2A, Int. J. Adv. Comput. Sci. Appl. (2021) 523–529.
[3] W. Yang, D. Pingping, C. Lin, T. Wensheng, Loss-aware throughput estimation scheduler for multipath TCP in heterogeneous wireless networks, IEEE Trans. Wirel. Commun. (2021) 3336–3349.
[4] K. Vidya, S. Sudhir, Exploring multipath TCP schedulers in heterogeneous networks, Int. J. Inf. Commun. Technol. Hum. Dev. (2022) 1–11.
[5] X. Yitao, K. Xue, Y. Zhang, J. Han, L. Jian, L. Jianqing, L. Ruidong, A low-latency MPTCP scheduler for live video streaming in mobile networks, IEEe Trans. Wirel. Commun. (2021) 7230–7242.
[6] B. Kimura, L. Demetrius, L. Antonio, Packet scheduling in multipath TCP: fundamentals, lessons, and opportunities, IEEE Syst. J. (2020) 1445–1457.
[7] W. Hongjia, A. Ozgu, B. Anna, F. Simone, C. Giuseppe, Peekaboo: learning-based multipath scheduling for dynamic heterogeneous environments, IEEE J. Select. Areas Commun. 38 (10) (2020) 2295–2310.

[8] A. Shivang, S. Swetank, I. Khan, P. Rohan, K. Dimitrios, W. Joerg, MuSher: an agile multipath-TCP scheduler for dual-band 802.11ad/ac wireless LANs, IEEE/ACM Trans. Network. 30 (2022) 1879–1894.

[9] D. Enhuan, M. Xu, F. Xiaoming, C. Yu, A Loss aware MPTCP scheduler for highly lossy networks, Comput. Netw. J. 157 (2019) 146–158.

[10] S. Ferlin, A. Ozgu, M. Olivier, B. Roksana, BLEST: blocking estimation-based MPTCP scheduler for heterogeneous networks, in: FIP Networking Conference (IFIP Networking) and Workshops, 2016, pp. 431–439.

[11] K. Vidya, S. Sawarkar, Evaluation of MPTCP schedulers in diverse scenarios, Eur. J. Electric. Eng. Comput. Sci. 5 (2021) 50–54.

[12] S. Hang, C. Yong, X. Wang, Y. Hu, M. Dai, W. Wang, K. Zheng, STMS: improving MPTCP Throughput under Heterogeneous Networks, in: USENIX Annual Technical Conference, 2018, pp. 719–730.

[13] L. Yeon, N. Erich, T. Don, G. Richard, ECF: an MPTCP path scheduler to manage heterogeneous paths, in: 13th International Conference on emerging Networking EXperiments and Technologies, 2017, pp. 147–159.

[14] C. Rajnish, S. Chand, An adaptive and efficient packet scheduler for multipath TCP, Iran. J. Sci. Technol., Trans. Electric. Eng. 45 (2020) 349–365.

[15] L. Tuan, L. Bui, Forward delay-based packet scheduling algorithm for multipath TCP, Mobile Netw. Appl. J. 23 (2018) 4–12.

[16] J. Han, K. Xue, X. Yitao, L. Jian, W. Wenjia, D. Wei, X. Guoliang, Leveraging coupled BBR and adaptive packet scheduling to boost MPTCP, IEEe Trans. Wirel. Commun. (2021) 1–14.

[17] B. Han, F. Qian, J. Lusheng, V. Gopalakrishnan, MP-DASH: adaptive video streaming over preference-aware multipath, in: 12th International on Conference on Emerging Networking EXperiments and Technologies, 2016, pp. 129–143.

[18] D. Pingping, J. Xie, T. Wensheng, X. Naixue, H. Zhong, V. Athanasios, Performance Evaluation of Multipath TCP Scheduling Algorithms, IEEE Access, 2019, pp. 29818–29825.

[19] K. Nicolas, L. Emmanuel, M. Ahlem, S. Golam, M. Olivier, B. Roksana, DAPS: intelligent delay-aware packet scheduling for multipath transport, in: IEEE International Conference on Communications, 2014, pp. 4–12.

[20] P. Kumar, F. Nida, P. Saxena, Performance analysis of multipath transport layer schedulers under 5G/B5G Hybrid Networks, in: 14th International Conference on Communication Systems, 2022, pp. 658–666.

[21] N. Thakur, K. Ashwini, Analysing schedulers of multipath TCP in diverse environment, in: International Conference on Advances in Computing, Communication Control and Networking (ICACCCN), 2021, pp. 1337–1340.

[22] S. Nagayama, C. Dirceu, N. Daiki, I. Takeshi, Path switching schedulers for MPTCP Streaming Video, IEEE Pacific Rim Conf. Commun., Comput. (2019) 1–6.

[23] L. Ralf, W. Philip, G. Sascha, RTTPROBS: a RTT probing scheme for RTT aware multipath scheduling, in: IEEE 93rd Vehicular Technology Conference (VTC2021-Spring), 2021, pp. 1–6.

[24] S. Muge, E. Karayer, P. Chi, S. Stefano, B. Selma, Numerical Evaluation of MPTCP schedulers in terms of throughput and reliability. 11th International Workshop on Resilient Networks Design and Modeling (RNDM), 2019, pp. 1–6.

[25] N. Kensuke, I. Yoshihiro, Proposal of Multipath TCP packet scheduler to adjust trade-off between QoS Fluctuation and throughput for WebQoE improvement, in: fourth International Conference on Computer and Communication Systems, 2019, pp. 493–496.

[26] L. Li, K. Xu, L. Tong, K. Zheng, C. Peng, D. Wang, X. Wang, M. Shen, R. Mijumbi, A measurement study on multipath TCP with multiple cellular carriers on high speed rails, in: Conference of the ACM Special Interest Group on Data Communication (SIGCOMM '18), 2018, pp. 161–175.

[27] S. Habib, Q. Junaid, A. Anwaar, H. Durdana, L. Ming, S. Arjuna, The past, present, and future of transport-layer multipath, J. Netw. Comput. Appl. 75 (2016) 236–258.

[28] C. Quentin, B. Matthieu, H. Benjamin, B. Olivier, Observing real smartphone applications over multipath TCP, IEEE Commun. Mag. 54 (2016) 88–93.

[29] B. Han, F. Qian, S. Hao, J. Lusheng, An anatomy of mobile web performance over multipath TCP, in: 11th ACM Conference on Emerging Networking Experiments and Technologies, 2015, pp. 1–7.

[30] K.J. Grinnemo, A. Brunstrom, A first study on using MPTCP to reduce latency for cloud based mobile applications, in: IEEE Symposium on Computers and Communication (ISCC), 2015, pp. 64–69.

[31] P. Houze, M. Emmanuel, G. Texier, S. Gwendal, Applicative-layer multipath for low-latency adaptive live streaming, in: IEEE International Conference on Communications (ICC), 2016, pp. 1–7.

[32] M. Li, L. Andrey, O. Zhonghong, Y. Antti, T. Sasu, C. Matthieu, S. Stefano, Multipath transmission for the internet: a survey, IEEE Commun. Surv. Tutor. 18 (2016) 2887–2925.

[33] M. Polese, C. Federico, B. Elia, R. Filippo, Z. Andrea, Z. Michele, A survey on recent advances in transport layer protocols, IEEE Commun. Surv. Tutor. 21 (4) (2019) 3584–3608.

[34] M. Morawski, P. Ignaciuk, Energy efficient MPTCP transmission-scheduler implementation and evaluation, in: 21st International Conference on System Theory, Control and Computing (ICSTCC), 2017, pp. 654–659.

[35] E. Dong, M. Xu, X. Fu, Y. Cao, LAMPS: a loss aware scheduler for multipath tcp over highly lossy networks, in: IEEE 42nd Conference on Local Computer Networks (LCN), 2017, pp. 1–9.

[36] H. Zhang, W. Li, S. Gao, X. Wang, B. Ye, ReLeS: a neural adaptive multipath scheduler based on deep reinforcement learning, IEEE Conf. Comput. Commun. (2019) 1648–1656.

[37] Y. Xing, K. Xue, Y. Zhang, J. Han, J. Li, D.S.L. Wei, An online learning assisted packet scheduler for MPTCP in mobile networks, IEEE/ACM Trans.Netw. (2023) 1–16.

[38] Q. Peng, A. Walid, J. Hwang, S. Low, Multipath TCP: analysis, design, and implementation, IEEE/ACM Trans. Netw. 24 (1) (2016) 596–609.

[39] S. Afzal, V. Testoni, E. Rothenberg, C. Kolan, B. Imed, A holistic survey of multipath wireless video streaming, J. Network and Comput. Appl. 212 (2023).

[40] Q.D. Coninck, O. Bonaventure, Multipath QUIC: design and evaluation, in: 13th International Conference on Emerging networking Experiments and Technologies, 2017, pp. 160–166.

[41] D. Coninck, O. Bonaventure, Multipathtester: comparing MPTCP and MPQUIC in mobile environments, in: Network Traffic Measurement and Analysis Conference (TMA), 2019, pp. 221–226.

[42] T. Viernickel, A. Froemmgen, A. Rizk, B. Koldehofe, R. Steinmetz, Multipath QUIC: a deployable multipath transport protocol, in: IEEE International Conference on Communications (ICC), 2018, pp. 1–7.

[43] Q.D. Coninck, O. Bonaventure, Multiflow QUIC: a generic multipath transport protocol, IEEE Commun. Mag. 59 (5) (2021) 108–113.

[44] S. Przylucki, D. Czerwinski, The simulation study on the multipath adaptive video transmission, in: International Conference of Computational Methods in Engineering Science (CMES'18) 252, 2019.

[45] W. Tang, Y. Fu, P. Dong, W. Yang, B. Yang, N. Xiong, A MPTCP scheduler combined with congestion control for short flow delivery in signal transmission, In IEEE Access 7 (2019) 116195–116206.

[46] A. Mondal, A.R. Kabbinale, S. Shailendra, H.K. Rath, A. Pal, PPoS: a novel sub-flow scheduler and socket APIs for multipath TCP (MPTCP), in: Twenty Fourth National Conference on Communications (NCC), 2018, pp. 1–6.

[47] R. Helmke, S. Thieme, B. Schuetz, Improving connectivity in multipath PLMN setups: an MPTCP scheduler using link quality indicators, in: Mobile Communication-Technologies and Applications; 26th ITG-Symposium, 2022, pp. 1–5.

[48] V. Yedugundla, F. Simone, D. Thomas, A. Ozgu, K. Nicolas, H. Per, B. Anna, Is multipath transport suitable for latency sensitive traffic? Comput. Netw. 105 (2016).

[49] R. Matsufuji, S. Nagayama, D. Cavendish, D. Nobayashi, T. Ikenaga, TCP state driven MPTCP packet scheduling for streaming video, in: IARIA 10th International Conference on Evolving Internet, 2018, pp. 9–14.

[50] F. Silva, D. Bogusevschi, G. Muntean, A MPTCP-based RTT aware packet delivery prioritization algorithm in AR/VR scenarios, in: Proceedings of IEEE Intern. Wireless Communications & Mobile Computing Conference (IWCMCC) 18, 2018, pp. 95–100.

[51] C. Sysomphone, P. Pattarawit, K. Chatchai, The path scheduling for MPTCP end-to-end hosts, in: 14th International Conference on Electrical Engineering, 2017, pp. 127–130.

[52] T. Shreedhar, N. Mohan, K. Sanjit, J. Kaul, Kangasharju, QAware: a cross-layer approach to MPTCP scheduling, in: Proceedings of International on Conference on Networking and Workshops, 2018, pp. 1–9.

[53] K. Xue, J. Han, D. Ni, W. Wenjia, C. Ying, X. Qing, H. Peilin, DPSAF: forward prediction based dynamic packet scheduling and adjusting with feedback for multipath TCP in lossy heterogeneous networks, IEEe Trans. Veh. Technol. (2017) 1521–1534.

[54] A. Frommgen, J. Heuschkel, B. Koldehofe, Multipath TCP scheduling for thin streams: active probing and one-way delay-awareness, in: IEEE International Conference on Communications (ICC), 2018, pp. 1–7.

[55] B.L. Kimura, D.F. Lima, A. Loureiro, Alternative scheduling decisions for multipath TCP, In IEEE Commun. Lett. 21 (11) (2017) 2412–2415.

[56] M. Morawski, P. Ignaciuk, Synchronizing scheduler for MPTCP transmission of streaming content, in: IEEE International Conference on Systems, Man, and Cybernetics (SMC), 2022, pp. 909–914.

[57] L. Chen, T. Wensheng, D. Pingping, Y. Wenjun, L. Xiaoping, Z. Hangjun, Blocking time-based MPTCP scheduler for heterogeneous networks, in: 4th International Conference on Cloud Computing and Security, 2018, pp. 364–375.

[58] K. Fahmi, D.J. Leith, S. Kucera, H. Claussen, SOS: stochastic object-aware scheduler for low delay communication over multiple wireless paths, in: IEEE International Conference on Communications (ICC), 2020, pp. 1–6.

[59] K.W. Choi, Y.S. Cho, J.W. Lee, S.M. Cho, J. Choi, Optimal load balancing scheduler for MPTCP-based bandwidth aggregation in heterogeneous wireless environments, Comput. Commun. 112 (2017) 116–130.

[60] P. Hurtig, G. Karl, B. Anna, F. Simone, A. Ozgu, K. Nicolas, Low-latency scheduling in MPTCP, IEEE/ACM Trans. Netw. 27 (2018) 302–315.

[61] K. Gao, C. Xu, J. Qin, L. Zhong, G. Muntean, A stochastic optimal scheduler for multipath TCP in software defined wireless network, in: IEEE International Conference on Communications (ICC), 2019, pp. 1–6.

[62] W. Lu, D. Yu, M. Huang, B. Guo, PO-MPTCP: priorities-oriented data scheduler for multimedia multipathing services, Int. J. Dig. Multimedia Broadc. (2018) 1–9.

[63] C. Paasch, S. Ferlin, O. Alay, O. Bonaventure, Experimental evaluation of multipath TCP schedulers. ACM SIGCOMM Workshop on Capacity Sharing, 2014, pp. 27–32.

[64] W. Wenjia, L. Xue, J. Han, W. David, H. Peilin, Shared bottleneck-based congestion control and packet scheduling for multipath TCP, IEEE/ACM Trans. Netw. (2020) 1–14.

[65] R. Neha, A. Thakur, S. Kunte, Efficient architecture with a general optimized redundant error based MPTCP scheduler, ICTACT J. Commun. Technol. 13 (2) (2022) 2694–2705.

[66] F. Yang, P. Amer, N. Ekiz, A scheduler for multipath TCP, in: 22nd International Conference on Computer Communication and Networks (ICCCN), 2013, pp. 1–7.

[67] P. Dong, W. Yang, W. Tang, J. Huang, H. Wang, Y. Pan, J. Wang, Reducing transport latency for short flows with multipath TCP, J. Netw. Comput. Appl. 108 (2018) 20–36.

Atefeh Ahmadniai Khajekini received the BSc degree in Electrical and Electronic Engineering from the Islamic Azad University of Lahijan (Iran) in 2008. In October 2013, she joined the Department of Computer Engineering at the University of Guilan, Iran, as a full-time MSc student. She received the MSc degree in 2015. She is a Ph.D. student under advisor Ali Hakan Ulusoy in the Electrical and Electronic Engineering Department of Eastern Mediterranean University (EMU). She is a senior instructor with the Electrical and Electronic Engineering Department at the Final International University in Northern Cyprus. Her research interests include computer networks, wireless networks, network security, cryptography algorithms, and mobile communications.

Prof. Dr. Hasan AMCA received his B.S. degree in Electrical and Electronic Engineering from the Eastern Mediterranean University (formerly called Higher Technological Institute) in the North Cyprus in 1984, the M.Sc. degree from the University of Essex (UK) in 1985 and the Ph.D. degree from the University of Bradford (UK) in 1993. He works as a lecturer in the Electrical and Electronic Engineering Department of the EMU. He served as the Director of the School of Computing and Technology, the dean of Engineering Faculty, director of Continuing Education Center and Vice Rector responsible for International Relations and Recruitment at EMU. He also served as the Chair of Board of Trustees of Ataturk Teachers Academy, member of the Information and Communication Technology Regulatory Board (BTHK) in North Cyprus and Chair of the Board of directors of Cyprus Turkish Electricity Authority in May-June 2022. He also served as consultant to ministry of Health and ministry of communication and transport. Prof. Dr. Hasan AMCA is a senior member of the IEEE. His current research interests are channel estimation and detection over multipath propagation medium for 3G/4G/5G/ Mobile Communications Systems, OFDM, LTE, Millimeter-Wave Communications, Digital Video Broadcasting and Mobile Payment Systems.

Prof. Dr. Ali Hakan Ulusoy was born on June 3, 1974, in Eskişehir, Turkey. He graduated from the double major program of the Department of Electrical and Electronic Engineering (EEE) and the Department of Physics at Eastern Mediterranean University (EMU) in 1996. Subsequently, he earned his M.S. and Ph.D. degrees in EEE from EMU in 1998 and 2004, respectively. In 2004, he joined the Department of Information Technology at EMU as an instructor. Over the years, he progressed through various academic ranks: he became a visiting assistant professor in 2005, an assistant professor in 2007, an associate professor in 2013, and finally, a professor in 2019. During his tenure at EMU, he held administrative roles at the Institute of Graduate Studies and Research (IGSR). He served as the assistant director from January 2012 to August 2017 and as the acting director from August 2017 to December 2019. Since January 2020, he has been serving as the director of IGSR. His research interests span across various areas, including wireless communications, receiver design, channel estimation, fuzzy systems, wireless networks, cloud computing, millimeter-wave communications, healthcare system development, and support vector machines for intrusion detection in vehicular ad-hoc networks.

Prof. Dr. ENVER EVER received an M.Sc. degree in computer networks and a PhD in performance evaluation of computer networks and communication systems from Middlesex University, London, in 2004 and 2008, respectively. He was a Post-Doctoral Research Associate with Bradford University for one year. He was a Lecturer/Senior Lecturer with the Computer and Communications Engineering Department at Middlesex University from 2008 to 2013. He is a Professor at the Middle East Technical University Northern Cyprus Campus. His current research interests include the Internet of Things, computer networks, wireless sensor networks, wireless multimedia sensor networks, artificial intelligence (machine learning/deep learning) applications, wireless communication systems, cloud computing, and performance/reliability modelling. He serves on various boards and program committees.