



# ADAS: Adaptive Delay-Aligned Scheduling for Multipath Transmission in Heterogeneous Wireless Networks

Du Chen<sup>1</sup> · Deyun Gao<sup>1</sup> · Lu Jin<sup>1</sup> · Wei Quan<sup>1</sup> · Hongke Zhang<sup>1</sup>

Received: 12 July 2022 / Accepted: 21 February 2023 / Published online: 3 April 2023  
© The Author(s), under exclusive licence to Springer Science+Business Media, LLC, part of Springer Nature 2023

## Abstract

Multipath TCP (MPTCP) is regarded as a promising solution to aggregate the bandwidth of multiple paths to generate throughput benefits in heterogeneous networks. However, transmitting data over multiple paths simultaneously often leads to out-of-order issues due to the asymmetry of heterogeneous paths. In this paper, we propose a novel packet scheduling mechanism named Adaptive Delay-Aligned Scheduling (ADAS) for multipath transmission in heterogeneous wireless networks. ADAS utilizes the wisdom of the last-hop connected to the receiver to solve the out-of-order problem and improve the overall throughput at the same time. Specifically, ADAS equips a virtual link loop on the last-hop to buffer the out-of-order packets within time threshold and further schedules and sends them to the receiver as sequentially as possible, which looks as if all packets take the same time to travel across the network. In this way, the delay-aligned scheduling is achieved and the out-of-order problem can be effectively addressed. Besides, an adaptive weighting algorithm is proposed to dynamically adjust the time threshold to avoid over scheduling and improve the overall throughput. Extensive experiments demonstrate that ADAS outperforms state-of-the-art mechanisms. Besides, a lower out-of-order rate of 5.73% and a higher overall throughput of 6.76 Mbps can be achieved through combining ADAS with the current scheduling mechanisms.

**Keywords** Heterogeneous wireless networks · Multipath transmission · Out-of-order · Virtual link loop

## 1 Introduction

The rapid development of telecommunication business and the continuous emergence of new applications have put forward higher requirements for bandwidth [1]. With the advent of multiple network interfaces on devices [2], aggregating the bandwidth of multiple paths to generate throughput benefits has become a natural evolution. However, the

traditional Transmission Control Protocol (TCP) allows only single path to access the Internet at a time [3]. As a promising solution, Multipath TCP (MPTCP) extends TCP, enabling the devices with multiple network interfaces to transmit data over multiple paths simultaneously [4]. Supported by MPTCP, a higher end-to-end throughput can be potentially achieved and the robustness can be significantly improved during times of path failure.

Applying MPTCP to data transmission can effectively cope with the growing bandwidth demands. However, each heterogeneous path in MPTCP varies in transmission characteristics, including bandwidth, transmission delay, packet loss rate, and so on. Compared with single path in TCP, the asymmetric heterogeneous paths in MPTCP will cause more out-of-order packets, which will block the receive buffer and degrade the transmission performance [5, 6]. Since the out-of-order problem is more prominent in multipath transmission scenarios, it has attracted widespread research interest in both academia and industry. A variety of solutions have been proposed to overcome the out-of-order problem [7], most of which are sender-centric. In such solutions, the receiver is restricted to acknowledging receipt of packets and using acknowledgment

---

✉ Deyun Gao  
gaody@bjtu.edu.cn  
Du Chen  
duchen@bjtu.edu.cn  
Lu Jin  
jinlu1@bjtu.edu.cn  
Wei Quan  
weiquan@bjtu.edu.cn  
Hongke Zhang  
hkzhang@bjtu.edu.cn

<sup>1</sup> School of Electronic and Information Engineering, Beijing Jiaotong University, No. 3 Shangyuancun, Beijing 100044, China

chunks to provide feedback, while the sender executes all key operations based on the feedback. However, the network state may change several times before the sender receives the feedback such that the operations of the sender cannot fit the current network state [8]. Furthermore, the situation can be even worse in wireless heterogeneous networks due to frequent packet loss [9]. Besides, to enable these solutions, customized changes on the sender and/or the receiver are required, which is relatively complicated. Therefore, these solutions cannot effectively solve the out-of-order problem and fully utilize the bandwidth of multiple paths.

In this paper, we propose a novel packet scheduling mechanism named Adaptive Delay-Aligned Scheduling (ADAS), which is a fresh attempt to consider both out-of-order mitigation and throughput improvement for data transmission in heterogeneous wireless networks. Different from the existing designs, ADAS schedules and processes the packets on the last-hop connected the receiver. Firstly, a virtual link loop is equipped on the last-hop to buffer the out-of-order packets within time threshold  $T$ . Secondly, a packet scheduling algorithm is proposed to efficiently process the out-of-order packets based on the first-hand knowledge parsed from the packet header. At last, an adaptive weighting algorithm is proposed to dynamically adjust the time threshold  $T$  based on the connect state information. With an adaptive time threshold  $T$ , the over scheduling can be avoided and the overall throughput is improved. The main contributions of this paper are summarized as follows:

- We design ADAS for multipath transmission in heterogeneous wireless networks. ADAS utilizes the wisdom of the last-hop connected to the receiver and frees bandwidth aggregation from the out-of-order problem, leaving both the sender and the receiver unchanged.
- We propose an adaptive weighting algorithm to dynamically adjust the time threshold  $T$ . The overall throughput will decrease due to the out-of-order packets staying in the virtual link loop too long. An adaptive time threshold  $T$  can avoid over scheduling and improve the overall throughput.
- We establish a real-network environment and carry out extensive experiments to validate the effectiveness and superiority of ADAS. The results show that ADAS outperforms state-of-the-art mechanisms and allows parallel implementation with the current scheduling mechanisms to further promote the performance.

The remainder of this paper is organized as follows. Section 2 briefly reviews the related work. The core components and design principles of ADAS are introduced in details in Section 3. In Section 4, extensive experiments are carried out to validate the effectiveness and superiority of ADAS. Besides, the compatibility of ADAS is also validated

through combining ADAS with LowRTT and OTIAS. Finally, we conclude this paper in Section 5.

## 2 Related work

In the past few years, researchers have proposed various methods to solve the out-of-order problem in MPTCP. As a rule-of-thumb, expanding the receive buffer size to store more out-of-order packets is the most straightforward way to address this issue [10]. However, it does not reduce any out-of-order packets. Besides, the reordering of the out-of-order packets will cause a long latency, which is unacceptable for time-sensitive applications.

Some proposed mechanisms focus on the design of scheduling algorithms to ensure that the packets arrive at the receiver in order. Round-Robin (RR) is the simplest scheduling algorithm, which sends packets to each path in turn if the path has available window. RR can achieve a good performance in the scenario where the paths have homogeneous transmission characteristics. However, the paths are heterogeneous in most of the real-network scenarios. Using RR will cause a wealth of out-of-order packets.

To cope with the asymmetry of heterogeneous paths, many efforts have been devoted to optimize the scheduling algorithms. Currently, the latest MPTCP implementation (MPTCP v0.95 based on the Linux Kernel Longterm Support release v4.19) uses Lowest-RTT-First (LowRTT) [11, 12] as the default scheduling algorithm, which schedules packets based on the Round-Trip Time (RTT) estimation. It first sends packets to the path with the lowest RTT estimation, the path with next lower RTT estimation, and so forth. LowRTT performs well when transmitting short flows (size is in KB), the performance of which depends on transmission delay rather than bandwidth [13]. However, LowRTT does not consider other factors in the heterogeneous networks, such as congestion window, packet loss rate, and so on. The receiver may still receive out-of-order packets due to the different transmission characteristics of the asymmetric paths. Unlike LowRTT, some scheduling algorithms are developed based on multiple factors. Sarwar et al. proposed Delay Aware Packet Scheduling (DAPS) based on the awareness of per-path delay with respect to the combined overall capacity of the paths to proactively minimize the blocking inside receive buffer [14]. And Kuhn et al. developed an analytical model of maximum receive buffer blocking time to further extend DAPS [15]. Yang et al. proposed Out-of-order Transmission for In-order Arrival Scheduler (OTIAS) to mitigate jitter for time-sensitive applications [16]. Ferlin et al. pointed out that both DAPS and OTIAS cannot react upon network changes in a timely manner through extensive simulations and real-network experiments. They then proposed a send-window BLocking ESTimation scheduler (BLEST) to minimize HoL-blocking in heterogeneous networks [8]. In addition, Ke et al. proposed a multiple attribute-aware data

scheduling strategy for MPTCP (MPTCP-MA<sup>2</sup>) by taking RTT and congestion window into account and using an optimized path sorting algorithm to compare and sort all available paths [17]. Similar to MPTCP-MA<sup>2</sup>, Luo et al. also designed a new MPTCP multi-attribute aware data scheduling algorithm based on RTT and congestion state (RCDS) [18]. RCDS sends packets through the path with minimum RTT firstly for the RTT ratio times and then selects the path with the best congestion state once.

Although the above proposals can reduce the probability of packet out-of-order and improve overall throughput to a certain extent, they all ignore the influence of packet loss and suffer from significant performance degradation in lossy heterogeneous networks. To improve the robustness of the scheduling algorithms in lossy heterogeneous networks, Xue et al. proposed Forward Prediction based Dynamic Packet Scheduling and Adjusting with Feedback (DPSAF) to detect and reduce out-of-order packets by considering packet loss and using feedback information from SACK options for further correction [19]. However, DPSAF does not consider the scenario where MPTCP may suffer from retransmission timeout (RTO) in a highly lossy network. Dong et al. proposed Loss-Aware MPTCP Scheduler (LAMPS) for highly lossy networks, which considers both transmission delay and packet loss [20]. LAMPS relies heavily on the accuracy of the packet loss rate estimation. However, the packet scheduling process will influence the packet loss rate estimation when pushing many packets to a certain path. Hence, LAMPS cannot deal with loss rate switching as expected and suffers a performance degradation when burst packet losses happen. Yang et al. proposed Loss-Aware Throughput Estimation scheduler (LATE) through establishing a throughput estimation model, which comprehensively considers the transmission characteristics of each path including RTT, congestion window and packet loss rate [21]. With the throughput estimation model, LATE can estimate the data amount that each path can delivery in a certain round and schedule packets adaptively into different paths accordingly. The authors presented the experimental results of LATE under discrete increasing trends of packets loss rate, RTT, and filesize, which show that LATE outperforms BLEST and DPSAF. However, both RTT and packet loss rate may fluctuate irregularly, especially in heterogeneous wireless networks.

Recently, more and more novel mechanisms of Machine Learning (ML) based are used to solve the out-of-order problem in MPTCP due to the fusion of ML into the transport layer [22, 23]. Several works leverage Deep Reinforcement Learning to redesign scheduling algorithms for MPTCP [24–26]. However, there are no significant improvements compared to traditional algorithms. Li et al. proposed SmartCC and used an asynchronous reinforcement learning framework to learn a set of congestion rules. With these rules, the sender can observe the environment and take actions to adjust the congestion windows of the paths adaptively [27]. Wu et al. proposed Peekaboo which is aware of the dynamic transmission characteristics of the heterogeneous paths

[28]. However, the speed at which the mobile network changes may surpass the online learning speed of Peekaboo. Besides, there is not sufficient data for Peekaboo to learn the mobile network. Moreover, there are series of works applying ML to MPTCP for IoT applications [29, 30], industrial scenarios [31–33], and mobile devices [34, 35]. These proposals may work well in specific scenarios, but cannot be applied to others. In general, applying ML to MPTCP and utilizing ML to generate scheduling strategies face two challenges: (1) where to deploy the ML model and execute the training tasks, (2) the learning speed should be faster than the speed at which the network state changes. On the one hand, the end devices may have limited computing resources and are not capable of the training tasks. Executing the training tasks on the end devices is prone to lead a slow convergence, which causes the training results to be inapplicable to the current network state. On the other hand, despite offloading the training tasks to the remote servers with powerful computing capacity can profoundly improve the learning speed, it will introduce additional communication latency due to the parameter exchanging between the end devices and the remote servers. Only when these challenges are solved, can ML be better applied to MPTCP.

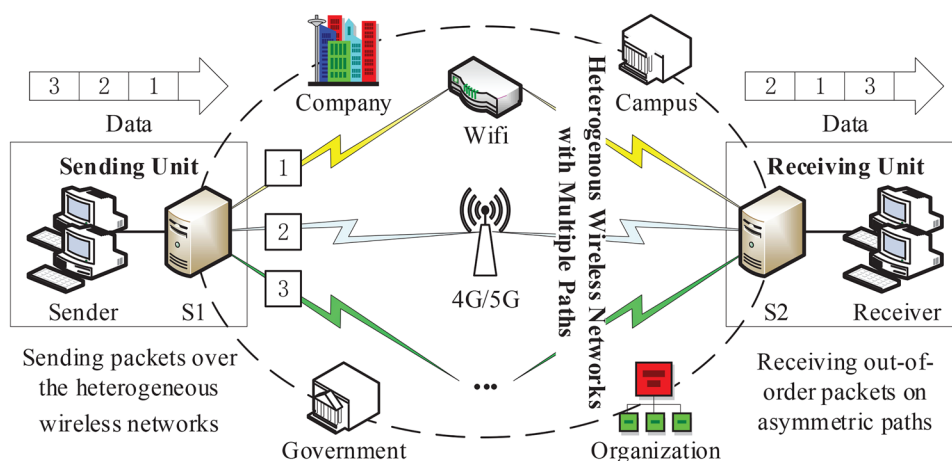
Based on the above discussion, we propose that multiple factors shall be considered, especially the packet loss, to overcome the out-of-order problem for MPTCP in heterogeneous wireless networks. Furthermore, the proposed mechanism shall be compatible to facilitate realistic deployment. Taking these into consideration, we propose ADAS to work on the last-hop connected to the receiver. On the one hand, ADAS can take full advantage of the first-hand knowledge parsed from the packet header, which can be easily obtained by the last-hop. On the other hand, ADAS can run in parallel with the current packet scheduling algorithms because it can be easily deployed on the network edge device and does not change any implementations of both the sender and the receiver. In fact, ADAS has better compatibility and robustness since it is implemented based on the data plane P4 language, which gives ADAS the potential to support various existing designs.

### 3 ADAS mechanism

ADAS is designed for data transmission in heterogeneous wireless networks. A typical scenario of heterogeneous wireless networks with multiple paths is illustrated in Fig. 1. The Sending Unit consists of the sender and the edge device S1. Similarly, the receiver and the edge device S2 compose the Receiving Unit. The Sending Unit transmits data to the Receiving Unit using a plurality of different access methods, including WIFI, 4G/5G, and so on.

ADAS mainly focuses on the Receiving Unit and is empowered with three core components, namely (1) virtual environment configuration, (2) packet scheduling algorithm, and (3) adaptive weighting algorithm. Specifically, a virtual link loop

**Fig. 1** A typical scenario of heterogeneous wireless networks with multiple paths



is configured on the edge device S2 to buffer the out-of-order packets within time threshold  $T$ . A packet scheduling algorithm is proposed to send the packets to the receiver as sequentially as possible to address the out-of-order problem. Besides, the time threshold  $T$  is dynamically adjusted to achieve a better overall throughput with the adaptive weighting algorithm.

### 3.1 Virtual environment configuration

Virtual network appliances make packet processing in the network edge devices more flexible, leaving both the sender and the receiver unchanged. For example, network namespaces provide isolation of the system resources associated with networking: network devices, IPv4 and IPv6 protocol stacks, IP routing tables, firewall rules, and so on. A physical network device can live in exactly one network namespace. A virtual Ethernet (veth) pair provides a pipe-like abstraction which can be used to create tunnels between network namespaces or bridges to physical network devices in another namespace. When a network namespace is freed, the physical network devices are moved back to the initial network namespace.

The edge device S2 isolates traffic in the virtual link loop by using a network namespace that shares a single set of network interfaces and routing table entries. To connect the current network namespace with the newly added network namespace, the veth interfaces need to be assigned to the network namespace. Suppose the edge device S2 has four physical interfaces, three of which are used to connect to the heterogeneous wireless networks with multiple paths, and the other one is used to connect to the receiver. As shown in Fig. 2, two veth pairs are added to connect different namespaces. The packets passing through port 4 will return to port 5 in the path of veth11 → veth12 → veth22 → veth21, which forms a virtual link loop. Here, we use the powerful Linux Traffic Control tools to delay veth12 for 1 ms to transmit packets to accomplish 1 ms delay of the virtual link loop. In this way, each out-of-order packet will take 1 ms to go through the virtual link loop every time. Therefore,

the delay-aligned scheduling with millisecond accuracy can be achieved as long as the out-of-order packets enter the virtual link loop for an appropriate number of times before being sent to the receiver.

### 3.2 Packet scheduling algorithm

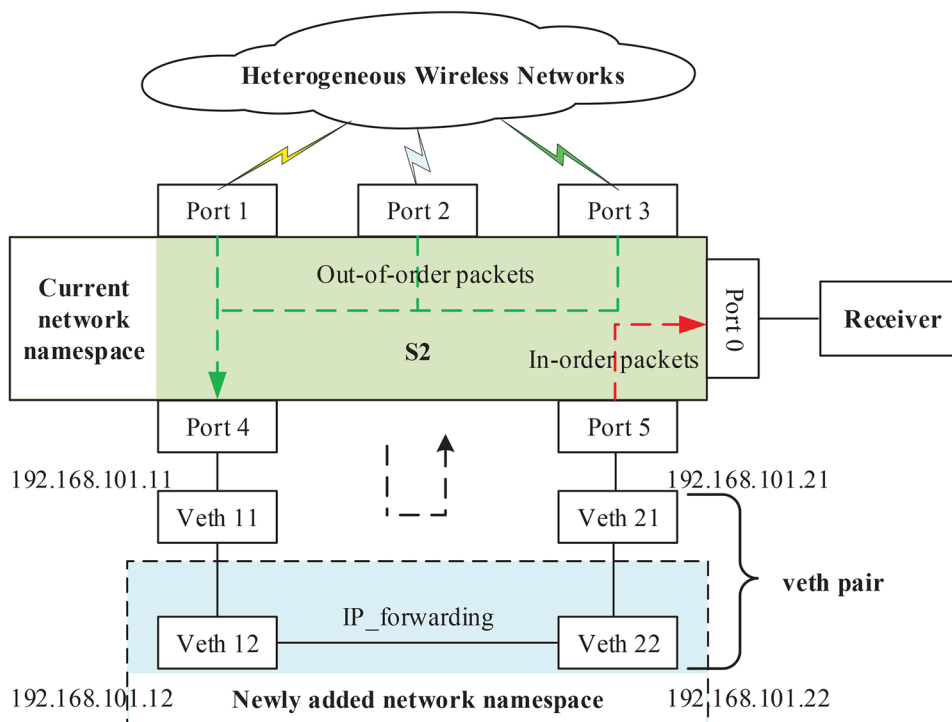
In heterogeneous wireless networks, the packets with smaller sequence number sent over a slower path may arrive at the edge device later than the packets with bigger sequence number sent over a faster path. Forwarding directly these packets from the edge device to the receiver would cause the receive buffer to be blocked by the out-of-order packets.

The edge device S2 can take full advantage of the first-hand knowledge parsed from the packet header to process the out-of-order packets as far as possible before forwarding them to the receiver. Specifically, the first-hand knowledge includes source IP address, destination IP address, source port number, destination port number and protocol (i.e., the five-tuples). In ADAS, Cyclic Redundancy Check 16 (CRC16) is used to hash the five-tuples of the packets so that each connection can be identified by the *hash* value with the connection information being masked. The calculation method of the *hash* value is expressed by Equation (1).

$$\text{hash}(\bullet) = \text{CRC16}(F) \bmod N \quad (1)$$

where  $F$  represents the contents of the five-tuples of the current packet, and  $N$  represents the size of the connection state register, which is a P4 programmable module. The larger the value of  $N$  is, the more space the register takes up, and the smaller probability of *hash* value conflict is. Considering the number of data streams and storage space in the real-network,  $N$  is set to 256, and the available interval of the index is 0x00–0xff. Therefore, the register contains 256 groups, and the size of each register does not exceed  $2^{32} - 1$ .

**Fig. 2** The virtual link loop configured on S2



Once a connection is well identified by the corresponding *hash* value, the sequence number of the expected next packet in this connection shall be calculated. To accomplish this task, the edge device S2 also collects the sequence number, the length of each protocol header, and other specific fields of the current packet so as to calculate the sequence number  $Es_i$  of the expected next packet in the  $i$ th connection. Taking the IPv4 protocol as an example, the abstract expression of the calculation method of  $Es_i$  is described as Equation (2).

$$Es_i = hdr.tcp.seq + hdr.ipv4.totalLen - hdr.ipv4.hdrLen - hdr.tcp.hdrLen \tag{2}$$

When a packet arrives, the edge device S2 will parse it and calculate the *hash* value to check whether the connection is newly established. If so, the sequence number  $Es_i$  of the expected next packet will be calculated and stored in the register with the *hash* value as the corresponding index. If not, the sequence number field will be extracted to compare with the corresponding  $Es_i$ . If the sequence number is greater than  $Es_i$ , the packet will be sent to the virtual link loop. Otherwise, the packet will be forwarded directly to the receiver with the  $Es_i$  stored in the register being updated. Once a packet  $P_i$  enters the virtual link loop, it will be continuously monitored. If the expected next packet arrives within time threshold  $T$ , it will be sent to the receiver together with packet  $P_i$ . Otherwise, if the process time exceeds time threshold  $T$  or the number of packets staying in the virtual link loop exceeds maximum number  $M$ , packet  $P_i$  will be sent directly to the receiver. The pseudo-code of packet processing is shown in Algorithm 1.

```

Algorithm 1 Packet Scheduling Algorithm


---


Input: Ingress port in_port of the packet
Time threshold T
Maximum number M
Output: Egress port eg_port of the packet
Initialize: count = 0
1: function TO_RECEIVER (header, hash)
2:   update the expected sequence number  $Es_i[hash]$ 
3:   eg_port = 0
4: end function
5: procedure PROCESSING THE PACKET (in_port)
6:   for every packet do
7:     if in_port == 1 || 2 || 3 then
8:       Calculate the hash value of the packet
9:       if hdr.tcp.seq >  $Es_i[hash]$  then
10:        count = count + 1
11:        if count < M then
12:          eg_port = 4
13:        else
14:          To_Receiver(header, hash)
15:        end if
16:      else
17:        To_Receiver(header, hash)
18:      end if
19:    else
20:      if process_time < T then
21:        Calculate the hash value of the packet
22:        if hdr.tcp.seq >  $Es_i[hash]$  then
23:          eg_port = 4
24:        else
25:          To_Receiver(header, hash)
26:          count = count - 1
27:        end if
28:      else
29:        To_Receiver(header, hash)
30:        count = count - 1
31:      end if
32:    end if
33:  end for
34: end procedure


---



```



Additionally, a scenario example of packet processing on the edge device S2 is depicted as Fig. 3. The packets are transmitted through two heterogeneous paths with different transmission delay. Since Path 1 is faster than Path 2, packet P5 arrives earlier at the edge device S2 than packet P4. According to the processing logic of ADAS, packet P5 is parsed and sent to the virtual link loop. After the arrival of packet P4, the two packets are delivered to the receiver in order. Besides, when packet loss occurs, such as packet P6, the next packet P7 will enter the virtual link loop. Once the process time exceeds time threshold  $T$ , the edge device S2 will directly forward packet P7 to the receiver. And the sender will send packet P6 again after receiving triple-duplicate ACKs or timeout.

With the packet scheduling algorithm, the last-hop delivers the packets to the receiver as sequentially as possible, which looks as if all packets take the same time to travel across the network. In this way, the delay-aligned scheduling is achieved and the out-of-order problem can be effectively addressed.

### 3.3 Adaptive weighting algorithm

The packet scheduling algorithm theoretically minimizes the probability of out-of-order conditions. However, due to the complexity of the heterogeneous wireless networks, the transmission characteristics of the heterogeneous paths may fluctuate frequently and packet loss may also occur. Simply setting

a larger time threshold  $T$  and letting the out-of-order packets stay in the virtual link loop for a fixed time will result in performance degradation caused by over scheduling. Therefore, the scheduling efficiency of ADAS is largely affected by the value of time threshold  $T$ . There must be an appropriate time threshold  $T$  to distinguish packet out-of-order and loss hole so that the packet scheduling algorithm can achieve a better performance, minimize the out-of-order rate and maximize the overall throughput. For these reasons, an adaptive weighting algorithm for adjusting the time threshold  $T$  is necessary.

The adaptive weighting algorithm is mainly for the scenario where the transmission characteristics of the heterogeneous paths fluctuate frequently and packet loss occurs at times. Generally, packet loss is detected by the reception of triple-duplicate ACKs or timeout in traditional TCP. The former will further trigger fast retransmit events. Padhye et al. have validated that there are much more retransmission timeout events than fast retransmit events [36]. In particular, MPTCP may suffer from retransmission timeout especially when transmitting short flows [37]. Besides, it is difficult to obtain the probabilities of fast retransmit events and retransmission timeout events. Therefore, we suppose that packet loss will trigger retransmission timeout each time when initialing the time threshold  $T$ .

Firstly, the propagation delay  $\tau_i$  considering retransmission of each path is calculated according to Equation (3).

$$\tau_i = (1 - loss_i) \frac{RTT_i}{2} + loss_i (RTO_i + \frac{RTT_i}{2}) \tag{3}$$

where  $loss_i$  represents the packet loss rate of the  $i$ th path.  $RTT_i$  and  $RTO_i$  represent the round-trip delay and retransmission time-out of the  $i$ th path, respectively. It should be noted that both the packet loss rate  $loss_i$  and the round-trip delay  $RTT_i$  of the  $i$ th path are measured multiple times in advance. Furthermore,  $RTO_i$  is calculated by Equations (4), (5), and (6) according to RFC 2988 [38].

$$RTO_i = SRTT_i + 4VAR_i \tag{4}$$

$$SRTT_i = (1 - 1/8)SRTT_{i-1} + 1/8RTT_i \tag{5}$$

$$VAR_i = (1 - 1/4)VAR_{i-1} + 1/4|SRTT_i - RTT_i| \tag{6}$$

Based on the propagation delay  $\tau_i$ , the average propagation delay difference  $\alpha$  of  $n$  heterogeneous paths can be calculated by Equation (7).

$$\alpha = \sum_{i=1}^n (\tau_i - \bar{\tau}_i) / n \tag{7}$$

Secondly, the propagation delay jitter is introduced to reflect the influence of the transmission characteristics fluctuation of the heterogeneous paths. Equation (8) is used to calculate the propagation delay jitter  $\beta$ .

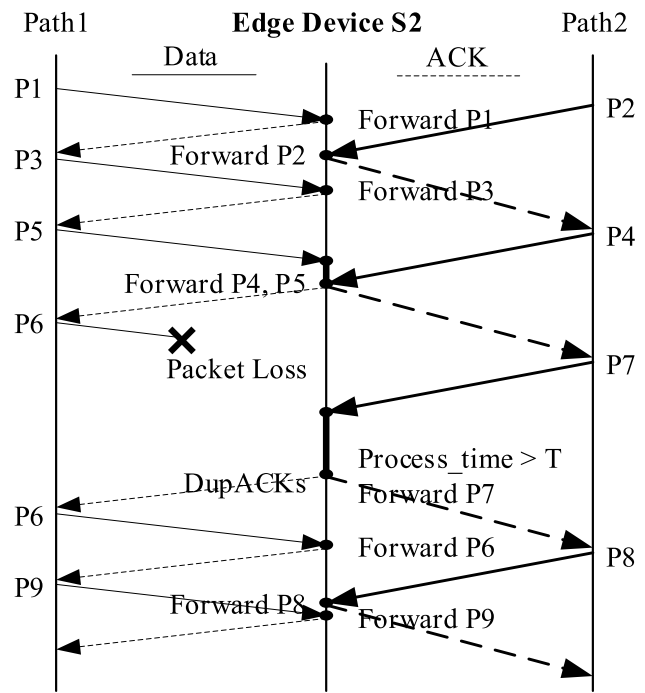


Fig. 3 A scenario example of packet processing on S2

$$\beta = \sqrt{\sum_{i=1}^n \left( \frac{rtt_i}{2} - \frac{\overline{rtt}}{2} \right)^2} / n \quad (8)$$

where  $rtt_i$  represents the round-trip delay of the  $i$ th packet while  $\overline{rtt}$  represents the average round-trip delay of  $m$  packets. Here, each round-trip delay  $rtt_i$  is also measured in advance. The average propagation delay jitter  $\beta$  represents the overall propagation delay jitter of heterogeneous wireless networks with multiple paths.

Given the definition of the average propagation delay difference  $\alpha$  and the average propagation delay jitter  $\beta$ , the initial value of time threshold  $T$  can be expressed as Equation (9).

$$T_{init} = \min(\alpha + \beta, 2\overline{rtt}) \quad (9)$$

Next, we focus on the dynamic adjustment of time threshold  $T$ . When the packet scheduling algorithm is running, labels can be used to distinguish packets that are processed differently in the program. Among all the packets forwarded from the virtual link loop to the receiver, the packets forwarded in order and the exception handling packets (including the packets that are forwarded directly to the receiver due to the process time exceeding time threshold  $T$  or the number of packets staying in the virtual link loop exceeding maximum number  $M$ ) are counted separately. The number of the former is denoted by  $N_o$  while the number of the latter is denoted by  $N_e$ . Therefore, the out-of-order improvement rate  $\eta$  can be calculated by Equation (10), which indicates the ratio of the number of out-of-order packets improved by the packet scheduling algorithm to the number of original out-of-order packets.

$$\eta = \frac{N_o}{N_o + N_e} \quad (10)$$

In addition to the out-of-order improvement rate, the overall throughput  $P$  is also used as a criterion to evaluate the time threshold  $T$ . When  $P$  is maximum, it means that the packet scheduling algorithm can maximize the overall throughput in heterogeneous wireless networks. Therefore, we use the product of the out-of-order improvement rate  $\eta$  and overall throughput  $P$  as the out-of-order optimization factor to evaluate the effectiveness of the packet scheduling algorithm. Equation (11) indicates the calculation method of the out-of-order optimization factor  $\varepsilon$ . The bigger  $\varepsilon$  is, the better the performance of the packet scheduling algorithm is.

$$\varepsilon = \eta P \quad (11)$$

Finally, with the definition of the out-of-order optimization factor  $\varepsilon$ , the edge device S2 can adjust the time threshold  $T$  according to Equation (12).

$$T(t+1) = \begin{cases} T(t) + \Delta, \theta > 0 \\ T(t) - \Delta, \theta \leq 0 \end{cases} \quad (12)$$

$$\theta = [\varepsilon(t) - \varepsilon(t-1)] \times [T(t) - T(t-1)]$$

where  $\Delta$  is the scheduling granularity while  $\theta$  reflects the reconcile utility of time threshold  $T$ .

The pseudo-code of the adaptive weighting algorithm is shown in Algorithm 2. During the packet scheduling process, ADAS measures and calculates the out-of-order optimization factor  $\varepsilon$  periodically and executes adaptive weighting according to Equation (12). In this way, ADAS improves the overall throughput while reducing the out-of-order rate.

## 4 Performance evaluation

In this section, extensive experiments are carried out and the performance of ADAS is comprehensively evaluated. Firstly, we set up an experimental platform and implement RR mechanism as a benchmark to validate the effectiveness of ADAS. Then, comparative experiments involving video stream transmission are carried out to compare ADAS with state-of-the-art mechanisms to demonstrate the superiority of ADAS. Finally, we combine ADAS with LowRTT and OTIAS to further promote the performance of ADAS.

---

### Algorithm 2 Adaptive Weighting Algorithm

---

**Input:** The initial value of time threshold  $T_{init}$

The scheduling granularity  $\Delta$

**Output:** Time threshold  $T$

**Initialize:**  $flag = 1$

```

1: procedure UPDATE  $T$  ( $T_{init}$ )
2:   while  $flag \neq 0$  do
3:     if  $flag == 1$  then
4:        $T[flag] = T_{init}$ 
5:        $\varepsilon[flag] = 0$ 
6:        $flag = flag + 1$ 
7:     else
8:        $\varepsilon[flag] = \eta P$ 
9:        $\theta = (\varepsilon[flag] - \varepsilon[flag - 1]) \times (T[flag] - T[flag - 1])$ 
10:      if  $\theta > 0$  then
11:         $T[flag] = T[flag] + \Delta$ 
12:         $flag = flag + 1$ 
13:      else
14:         $T[flag] = T[flag] - \Delta$ 
15:         $flag = flag + 1$ 
16:      end if
17:    end if
18:  end while
19: end procedure

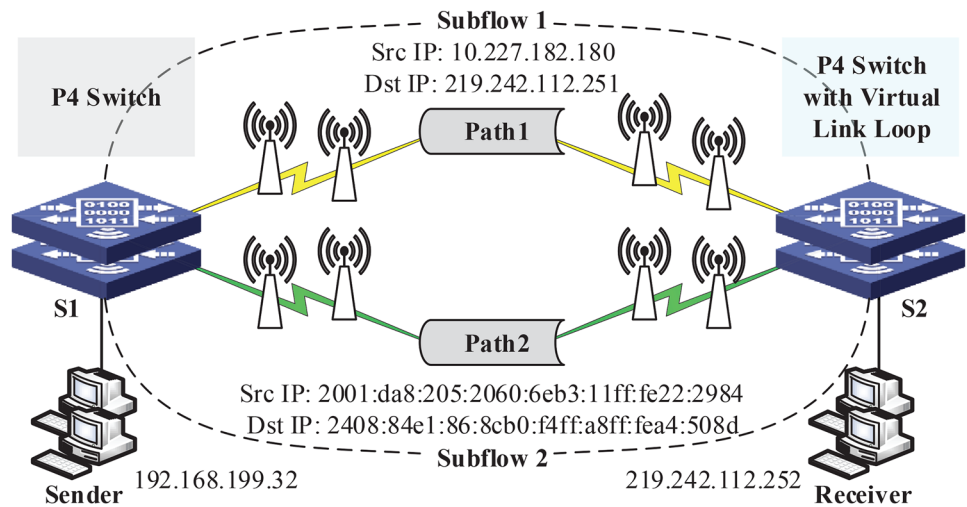
```

---

### 4.1 Experiment settings

The experimental topology is illustrated in Fig. 4. The sender and the receiver are connected to P4 switches S1 and S2, respectively. Specially, the switches are equipped with two 4G Subscriber Identification Module (SIM) cards, respectively.

**Fig. 4** The experimental network topology



One is assigned an IPv4 address while the other is assigned an IPv6 address. Besides, S2 is configured with a virtual link loop for packet scheduling and processing. The main parameters of the P4 switches are summarized in Table 1.

In the system, two paths are responsible for data transmission. The bandwidth and delay can be set by configuring the interface parameters of the switches. Both paths are wireless, and there are unavoidable delay jitter and packet loss. Besides, the actual throughput is generally slightly less than the available bandwidth. The maximum number  $M$  is used to indicate the total number of out-of-order packets that can stay in the virtual link loop. Theoretically, the total volume of traffic in fly is up to  $\sum_{i=1}^n b_i RTT_{max}$ , where  $b_i$  is the bandwidth of  $i$ th path and  $RTT_{max}$  is the highest  $RTT$  of all the paths. Besides,  $b_i$  and  $RTT_{max}$  are measured in bits per second and second, respectively. In the worst case, all these

bytes or packets arrive at S2 out-of-order. Therefore, the maximum number  $M$  is  $\sum_{i=1}^n b_i RTT_{max} / (8 \times 1500)$ . The scheduling granularity  $\Delta$  is used to adjust the time threshold  $T$  to let the out-of-order packets stay longer ( $T + \Delta$ ) or shorter ( $T - \Delta$ ) in the virtual link loop. Hence, it is reasonable to make the scheduling granularity  $\Delta$  equal to an integral multiple of the time to go through the virtual link loop once. Here, the scheduling granularity  $\Delta$  is set to 1 ms, which equals to the time to go through the virtual link loop once. In this way, the out-of-order packets will enter the virtual link loop one more time or one less time when the time threshold  $T$  is adjusted with the scheduling granularity  $\Delta$ .

**4.2 Validation of the effectiveness**

To validate the effectiveness of ADAS, we first use P4 to implement the RR mechanism. Then, Linux ethtool is used to configure the interface parameters of the switches to set different bandwidth and delay for the two paths. Here, we set the bandwidth of the Path 1 and Path 2 to be 5 Mbps and 3 Mbps, respectively. Besides, the delay of Path 1 is set to 50 ms while the delay of Path 2 is set to 100 ms. Iperf is used to send fixed size packets from the sender to the receiver. The out-of-order rate and overall throughput are measured to evaluate the effectiveness of ADAS. Through the statistics of packets captured on the receiver, the out-of-order rate is shown in Fig. 5a. The average out-of-order rate of RR is 36.95% while it decreases to 5.62% when ADAS is applied. With the efficiently scheduling of ADAS, the probability of packet out-of-order is greatly reduced, and it tends to get smaller and smaller in the later stage.

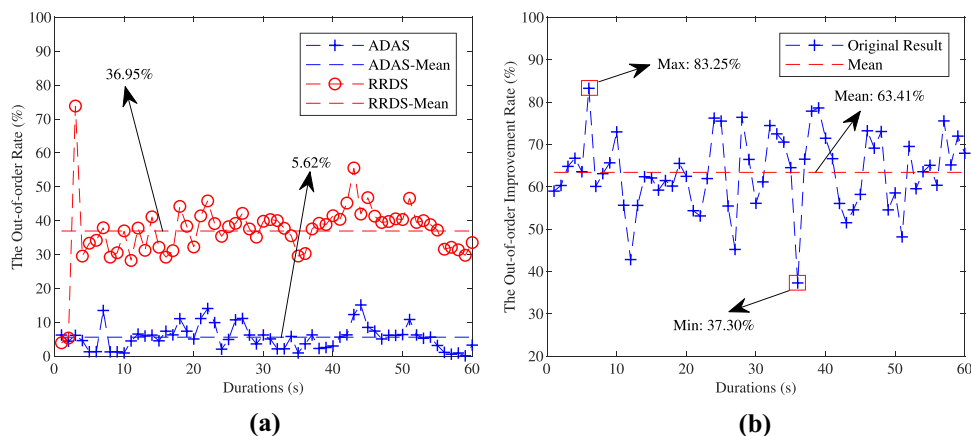
To further validate that ADAS can efficiently reduce the out-of-order packets, the out-of-order improvement rate is calculated and quantified. Specifically, all the packets forwarded from the virtual link loop (i.e., the edge device S2) to

**Table 1** Main Parameters of the P4 Switches

Parameters	S1	S2
CPU Model	Intel(R) Core(TM) i7-6500U CPU @ 3.20GHz	Intel(R) Xeon(R) CPU E5-2609 v4 @ 1.70GHz
CPU MHz	500.036	1360.066
Cache size	4096 KB	20480 KB
CPU Cores	8	8
MemTotal	3895676 kB	65851456 kB
Buffers	58268 kB	167432 kB
Cached	1301552 kB	643648 kB
Mac of NIC1	f6:ff:a8:a4:50:8d	6c:b3:11:22:29:84
Mac of NIC2	8e:92:96:ba:d6:33	6c:b3:11:22:29:85
IP of NIC1	10.227.182.180	219.242.112.251
IP of NIC2	2001:da8:205:2060:6eb3:11ff:fe22:2984	2408:84e1:86:8cb0:f4ff:a8ff:fea4:508d



**Fig. 5** The simulation results for ADAS in out-of-order mitigation: **a** The out-of-order rate; **b** The out-of-order improvement rate



the receiver are counted in conformance with two different processing logic. That is, the packets forwarded in order and the exception handling packets are counted separately. In this way, the out-of-order improvement rate can be calculated according to Equation (10). Fig. 5b shows that the average out-of-order improvement rate is 63.41% and the maximum out-of-order improvement rate reaches 83.25%. However, the minimum out-of-order improvement rate is only 37.30%. This is because the sudden packet loss leads to a large number of out-of-order packets. There is still a certain lag even if time threshold  $T$  is adjusted in time.

In addition, we also validate that ADAS can promote the overall throughput. As shown in Fig. 6, the average overall throughput of RR is about 2.58 Mbps, which is much smaller than the sum of the available bandwidth of the two paths. In the case of ADAS without the adaptive weighting algorithm (labeled as Unweighting ADAS), the average overall throughput is about 4.86 Mbps. By configuring a virtual link loop on the edge device S2, the out-of-order packets are scheduled and

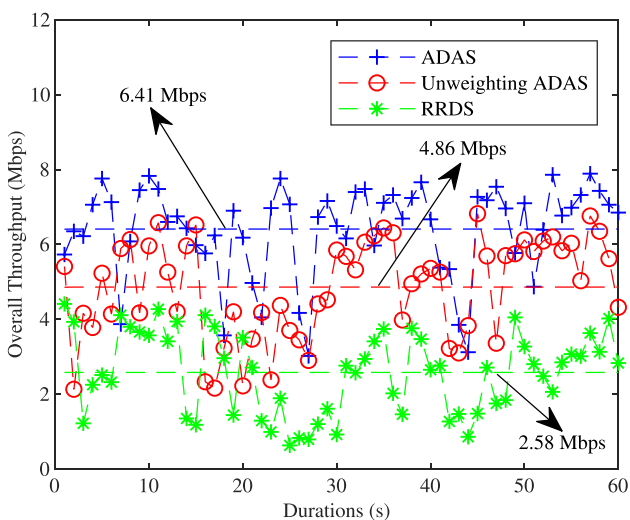
processed reasonably before being sent to the receiver. However, since the time threshold  $T$  is fixed and does not change with feedback, there is still a gap between the average overall throughput and the sum of the available bandwidth of the two paths. After using the adaptive weighting algorithm, the average overall throughput reaches 6.41 Mbps, which is approximately 2.52 times of that of RR. This fully shows that dynamically adjusting time threshold  $T$  can further improve the performance of ADAS and promote the overall throughput.

### 4.3 Comparison with state-of-the-art mechanisms

In this subsection, we mainly compare ADAS with the aforementioned three sender-centric scheduling mechanisms, namely LowRTT, OTIAS and LAMPS. To carry out the experiments, we deploy a video service on the sender and let the receiver access it. As shown in Fig. 7a and Fig. 7b, ADAS outperforms the three other mechanisms, achieving lowest average out-of-order rate and highest average overall throughput. Specifically, the average out-of-order rate of LowRTT, OTIAS, LAMPS and ADAS are 31.62%, 20.86%, 9.04% and 7.19%, respectively. And the average overall throughput of LowRTT, OTIAS, LAMPS and ADAS are 3.25 Mbps, 2.66 Mbps, 5.38 Mbps and 5.69 Mbps, respectively.

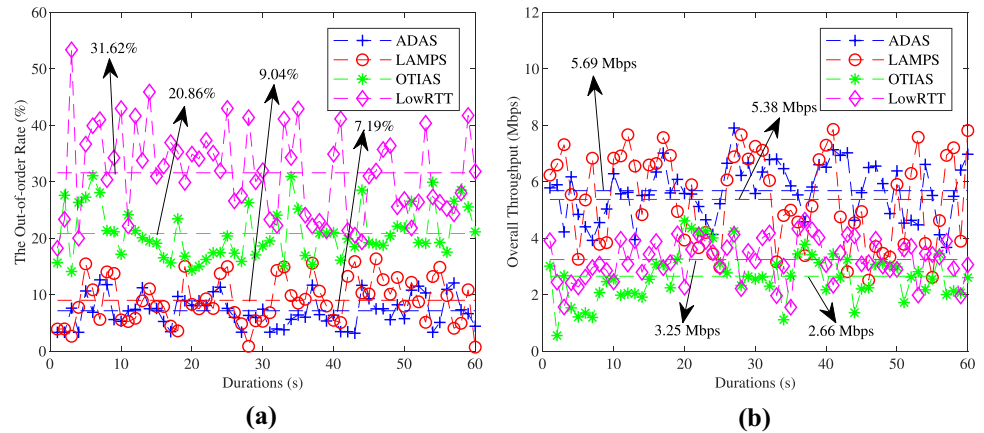
LowRTT simply allocates more packets to the path with lowest RTT, which means that the value of RTT dominates the performance of LowRTT. When the transmission characteristics of the paths are unstable or the RTT values are the same, LowRTT fails to work well. Besides, the path with lowest RTT is not always the best path to transmit data when the bandwidth of the path is very small. Pushing a large number of packets onto a path with a small bandwidth will only result in congestion and packet loss. Hence, the out-of-order rate of LowRTT is the highest among the four mechanisms.

Apart from LowRTT, OTIAS further considers the impact of congestion window size and introduces the concept of Delivery Delay (DeD) to indicate the time range between



**Fig. 6** The simulation results for ADAS in throughput improvement

**Fig. 7** The experimental results for comparing ADAS with state-of-the-art mechanisms: **(a)** The out-of-order rate; **(b)** The real-time throughput



when a packet is scheduled to a path and when that packet arrives in order at the receiver. On the basis of DeD, OTIAS transmits packets on different paths possibly out-of-order so that they arrive in order at the receiver. However, similar to LowRTT and other sender-centric scheduling mechanisms, the packets may not traverse the paths as expected once leaving the sender since the transmission characteristics of the paths fluctuate frequently and packet loss may occur in heterogeneous wireless networks. For this reason, OTIAS can only reduce the out-of-order rate to a certain degree but cannot completely eliminate the out-of-order packets. Besides, every packet may wait for several RTTs if the selected path has no available congestion window, which means that the volume of packets sent out by the sender in a certain time interval is restricted. As a result, the average overall throughput of OTIAS is the lowest among the four mechanisms.

LAMPS further considers packet loss in highly lossy networks, which selects the path with lowest transfer time and picks packet according to different path states (NORMAL state and REDUNDANT state). LAMPS can effectively reduce the out-of-order packets for video streaming traffic, reducing the extra bandwidth consumption while maintaining QoS. LAMPS shows better performance than LowRTT and OTIAS, whether in out-of-order mitigation or throughput improvement. However, both the computation of transfer time and the distinction between NORMAL state and REDUNDANT state rely heavily on the accuracy of the packet loss rate estimation. Pushing many packets to a certain path may influence the packet loss rate estimation because these packets may increase the probability of congestion when the capacity of the path is limited. Therefore, LAMPS fails to reduce the number of out-of-order packets as expected when burst packet loss happens.

Different from the sender-centric scheduling mechanisms, ADAS makes no changes on the sender but utilizes a virtual loop link to schedule and process packets on the last-hop connected to the receiver. According to the processing logic of the packet scheduling algorithm, the last-hop deliveries

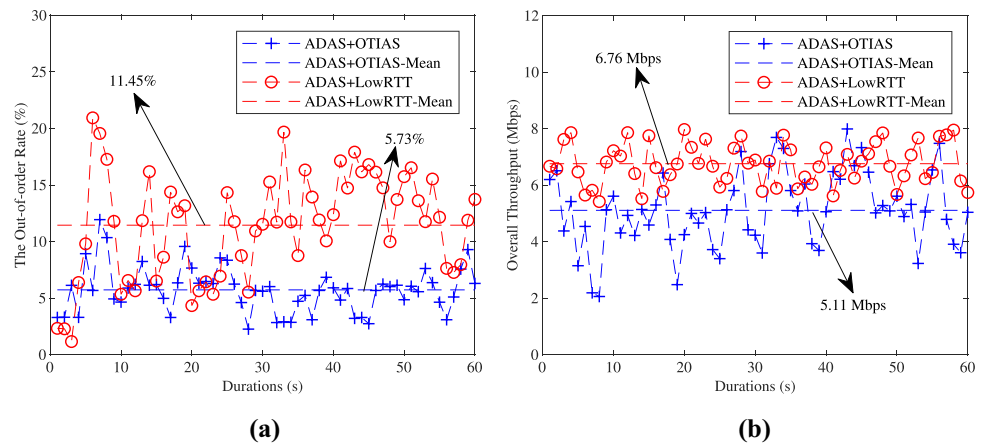
the packets to the receiver as sequentially as possible. In this way, all out-of-order packets processing is done directly before sending the packets to the receiver, which greatly eliminates the influence of transmission characteristics fluctuation of the paths. As a consequence, the final out-of-order packets are only the exception handling packets and the average out-of-order rate of ADAS is the lowest among the four mechanisms. In addition, ADAS uses the adaptive weighting algorithm to dynamically adjust time threshold  $T$  to avoid over scheduling. With a reasonable and adaptive time threshold  $T$ , the overall throughput can be further improved, which is the highest among the four mechanisms.

#### 4.4 Combination with LowRTT and OTIAS

We have validated the effectiveness and superiority of ADAS in the previous subsection. Next, we will combine ADAS with LowRTT and OTIAS respectively to validate that ADAS can be easily compatible with other scheduling mechanisms. Specifically, we use LowRTT and OTIAS to replace RRDS to send packets from the sender to the receiver and compare the performance of the combinations with that of the original ADAS (i.e., ADAS combined with RRDS). As shown in Fig. 8a, the average out-of-order rate reduces from 7.19% to 5.73% when combining ADAS with OTIAS while it increases from 7.19% to 11.45% when combining ADAS with LowRTT. In addition, Fig. 8b shows that the average overall throughput increases from 5.69 Mbps to 6.76 Mbps when combining ADAS with LowRTT while it reduces from 5.69 Mbps to 5.11 Mbps when combining ADAS with OTIAS.

On the one hand, different from the manner RRDS sends packets to each path in turn, OTIAS sends out packets out-of-order to ensure in order arrival at the receiver. Therefore, combining ADAS with OTIAS can further reduce the average out-of-order rate by 20.31% compared to that of the original ADAS. However, the packets may wait for several RTTs to send out when the path has no available congestion window, which results in the average overall throughput of combining

**Fig. 8** The experimental results for combining ADAS with LowRTT and OTIAS: **a** The out-of-order rate; **b** The real-time throughput



ADAS with OTIAS slightly reduces by 0.58 Mbps compared to that of the original ADAS.

On the other hand, LowRTT prefers the path with lowest RTT, and more packets can be sent by the sender in a certain time interval compared to RRDS. Hence, combining ADAS with LowRTT significantly increases the average overall throughput by 1.07 Mbps compared to that of the original ADAS. However, lots of out-of-order packets will generate when one packet is lost due to LowRTT continuously sending packets on the selected path until the congestion window is filled. The out-of-order packets contain not only the packets sent on the other path with bigger sequence number, but also the packets fall behind the lost one on the same path. Besides, the out-of-order problem will become more severe when the packet loss occurs on the slow path. For this reason, combining ADAS with LowRTT increases the average out-of-order rate by 59.25% compared to that of the original ADAS.

To sum up, ADAS outperforms state-of-the-art mechanisms and can easily work together with both LowRTT and OTIAS, which shows that ADAS has good compatibility. Unfortunately, compared to combining ADAS with RRDS, neither combining ADAS with LowRTT nor combining ADAS with OTIAS will improve overall throughput while reducing out-of-order packets. So we recommend choosing one of these combinations according to the requirements of specific application scenarios.

## 5 Conclusion

The order-of-order problem is a critical factor which influences the transmission performance of MPTCP, especially in heterogeneous wireless networks. This paper proposed ADAS to consider both out-of-order mitigation and throughput improvement in a more comprehensive fashion. ADAS took full advantage of the wisdom of the last-hop connected to the receiver. A virtual link loop was used to buffer the out-of-order packets within time threshold  $T$ , scheduling

and sending them to the receiver as sequentially as possible. Besides, an adaptive weighting algorithm was used to dynamically adjust the time threshold  $T$  to avoid over scheduling and further improve the overall throughput. Extensive experiments showed that ADAS not only outperforms but also is compatible with state-of-the-art mechanisms.

In the future, we will test ADAS in a more complex scenario where there are more than two heterogeneous paths. In addition, we will try to combine the latest sender-centric scheduling mechanisms to further improve the performance of ADAS.

**Author contributions** Du Chen wrote the main manuscript text and performed the analysis. Deyun Gao assisted in the analysis and manuscript preparation. Lu Jin established the testbed and carried out experiments. Wei Quan and Hongke Zhang provided constructive suggestions on the improvement of the manuscript. All authors reviewed the manuscript.

**Funding** This work is supported by the National Natural Science Foundation of China (grant no.61971028) and the National Key Research and Development Program of China (grant no.2018YFE0206800).

**Data availability** Not applicable.

## Declarations

**Ethical approval and consent to participate** Not applicable.

**Human and animal ethics** Not applicable.

**Consent for publication** Not applicable.

**Conflict of interest** The authors have no conflicts of interest to declare that are relevant to the content of this article.

## References

1. Wang D, Ding W, Ma X, Jiang H, Wang F, Liu J (2019) MiFo: a novel edge network integration framework for fog computing. Peer-to-peer Netw Appl 12(1):269–279. <https://doi.org/10.1007/s12083-018-0663-z>

2. Zhong L, Ji X, Wang Z, Qin J, Muntean GM (2022) A Q-learning driven energy-aware multipath transmission solution for 5G media services. *IEEE Trans Broadcast* 68(2):559–571. <https://doi.org/10.1109/TBC.2022.3147098>
3. Wang J, Liao J, Li T, Wang J (2015) On the collaborations of multiple selfish overlays using multi-path resources. *Peer-to-Peer Netw Appl* 8(2):203–215. <https://doi.org/10.1007/s12083-013-0245-z>
4. Ford A, Raiciu C, Handley M, Barre S, Iyengar J (2011) Architectural guidelines for multipath TCP development. <https://datacenter.ietf.org/doc/rfc6182/>
5. Morawski M, Ignaciuk P (2021) Choosing a proper control strategy for multipath transmission in industry 4.0 applications. *IEEE Trans Ind Inform* 18(6):3609–3619. <https://doi.org/10.1109/TII.2021.3105499>
6. Yu C, Quan W, Liu K, Liu M, Xu Z, Zhang H (2022) DRL-based fountain codes for concurrent multipath transfer in 6G networks. In: *Proceedings of 2022 IEEE Conference on Computer Communications Workshops (INFOCOM WKSHPS)*, New York, NY, USA, pp 1–6. <https://doi.org/10.1109/INFOCOMWKSHPS54753.2022.9798044>
7. Kimura BY, Lima DC, Loureiro AA (2020) Packet scheduling in multipath TCP: Fundamentals, lessons, and opportunities. *IEEE Syst J* 15(1):1445–1457. <https://doi.org/10.1109/JSYST.2020.2965471>
8. Ferlin S, Alay Ö, Mehani O, Boreli R (2016) BLEST: Blocking estimation-based MPTCP scheduler for heterogeneous networks. In: *Proceedings of 2016 IFIP Networking Conference (IFIP Networking) and Workshops*, Vienna, Austria, pp 431–439. <https://doi.org/10.1109/IFIPNetworking.2016.7497206>
9. Xu C, Li Z, Zhong L, Zhang H, Muntean GM (2015) CMT-NC: Improving the concurrent multipath transfer performance using network coding in wireless networks. *IEEE Trans Veh Technol* 65(3):1735–1751. <https://doi.org/10.1109/TVT.2015.2409556>
10. Xue K, Han J, Zhang H, Chen K, Hong P (2016) Migrating unfairness among subflows in MPTCP with network coding for wired-wireless networks. *IEEE Trans Veh Technol* 66(1):798–809. <https://doi.org/10.1109/TVT.2016.2543842>
11. Paasch C, Khalili R, Bonaventure O (2013) On the benefits of applying experimental design to improve multipath TCP. In: *Proceedings of the 9th ACM Conference on Emerging Networking Experiments and Technologies (CoNEXT)*, Santa Barbara, California, USA, pp 393–398. <https://doi.org/10.1145/2535372.2535403>
12. Paasch C, Ferlin S, Alay O, Bonaventure O (2014) Experimental evaluation of multipath TCP schedulers. In: *Proceedings of 2014 ACM SIGCOMM workshop on Capacity sharing workshop (CSWS)*, Chicago, Illinois, USA, pp 27–32. <https://doi.org/10.1145/2630088.2631977>
13. Kimura BY, Lima DC, Loureiro AA (2017) Alternative scheduling decisions for multipath TCP. *IEEE Commun Lett* 21(11):2412–2415. <https://doi.org/10.1109/LCOMM.2017.2740918>
14. Sarwar G, Boreli R, Lochin E, Mifdaoui A, Smith G (2013) Mitigating receiver's buffer blocking by delay aware packet scheduling in multipath data transfer. In: *Proceedings of 2013 27th International Conference on Advanced Information Networking and Applications Workshops (WAINA)*, Barcelona, Spain, pp 1119–1124. <https://doi.org/10.1109/WAINA.2013.80>
15. Kuhn N, Lochin E, Mifdaoui A, Sarwar G, Mehani O, Boreli R (2014) DAPS: Intelligent delay-aware packet scheduling for multipath transport. In: *Proceedings of 2014 IEEE International Conference on Communications (ICC)*, Sydney, NSW, Australia, pp 1222–1227. <https://doi.org/10.1109/ICC.2014.6883488>
16. Yang F, Wang Q, Amer PD (2014) Out-of-order transmission for in-order arrival scheduling for multipath TCP. In: *Proceedings of 2014 28th International Conference on Advanced Information Networking and Applications Workshops (WAINA)*, Victoria, BC, Canada, pp 749–752. <https://doi.org/10.1109/WAINA.2014.122>
17. Ke F, Huang M, Liu Z, Liu Q, Cao Y (2016) Multi-attribute aware multipath data scheduling strategy for efficient MPTCP-based data delivery. In: *Proceedings of 2016 22nd Asia-Pacific Conference on Communications (APCC)*, Yogyakarta, Indonesia, pp 248–253. <https://doi.org/10.1109/APCC.2016.7581457>
18. Luo J, Su X, Liu B, Zeng J (2018) Multi-attribute aware data scheduling for multipath TCP. In: *Proceedings of 2018 18th International Symposium on Communications and Information Technologies (ISCIT)*, Bangkok, Thailand, pp 270–274. <https://doi.org/10.1109/ISCIT.2018.8587933>
19. Xue K, Han J, Ni D, Wei W, Cai Y, Xu Q, Hong P (2017) DPSAF: Forward prediction based dynamic packet scheduling and adjusting with feedback for multipath TCP in lossy heterogeneous networks. *IEEE Trans Veh Technol* 67(2):1521–1534. <https://doi.org/10.1109/TVT.2017.2753398>
20. Dong E, Xu M, Fu X, Cao Y (2019) A loss aware MPTCP scheduler for highly lossy networks. *Comput Netw* 157:146–158. <https://doi.org/10.1016/j.comnet.2019.02.001>
21. Yang W, Dong P, Cai L, Tang W (2021) Loss-aware throughput estimation scheduler for multi-path TCP in heterogeneous wireless networks. *IEEE Trans Wirel Commun* 20(5):3336–3349. <https://doi.org/10.1109/TWC.2021.3049300>
22. Jiang H, Li Q, Jiang Y, Shen G, Sinnott R, Tian C, Xu M (2022) When machine learning meets congestion control: a survey and comparison. *Comput Netw* 192:108033. <https://doi.org/10.1016/j.comnet.2021.108033>
23. Siddiqi SJ, Naeem F, Khan S, Khan KS, Tariq M (2022) Towards AI-enabled traffic management in multipath TCP: a survey. *Comput Commun* 181:412–427. <https://doi.org/10.1016/j.comcom.2021.09.030>
24. Zhang H, Li W, Gao S, Wang X, Ye B (2019) ReLeS: a neural adaptive multipath scheduler based on deep reinforcement learning. In: *Proceedings of 2019 IEEE Conference on Computer Communications (INFOCOM)*, Paris, France, pp 1648–1656. <https://doi.org/10.1109/INFOCOM.2019.8737649>
25. Roselló MM (2019) Multi-path scheduling with deep reinforcement learning. In: *Proceedings of 2019 European Conference on Networks and Communications (EuCNC)*, Valencia, Spain, pp 400–405. <https://doi.org/10.1109/EuCNC.2019.8802063>
26. Yu C, Quan W, Gao D, Zhang Y, Liu K, Wu W, Zhang H, Shen X (2021) Reliable cybertwin-driven concurrent multipath transfer with deep reinforcement learning. *IEEE Internet Things J* 8(22):16207–16218. <https://doi.org/10.1109/JIOT.2021.3101447>
27. Li W, Zhang H, Gao S, Xue C, Wang X, Lu S (2019) SmartCC: a reinforcement learning approach for multipath TCP congestion control in heterogeneous networks. *IEEE J Sel Areas in Commun* 37(11):2621–2633. <https://doi.org/10.1109/JSAC.2019.2933761>
28. Wu H, Alay Ö, Brunstrom A, Ferlin S, Caso G (2020) Peekaboo: Learning-based multipath scheduling for dynamic heterogeneous environments. *IEEE J Sel Areas Commun* 38(10):2295–2310. <https://doi.org/10.1109/JSAC.2020.3000365>
29. Naeem F, Srivastava G, Tariq M (2020) A software defined network based fuzzy normalized neural adaptive multipath congestion control for the internet of things. *IEEE Trans Netw Sci Eng* 7(4):2155–2164. <https://doi.org/10.1109/TNSE.2020.2991106>
30. Ji R, Cao Y, Fan X, Jiang Y, Lei G, Ma Y (2020) Multipath TCP-based IoT communication evaluation: From the perspective of multipath management with machine learning. *Sensors* 20(22):6573. <https://doi.org/10.3390/s20226573>
31. Pokhrel SR, Garg S (2020) Multipath communication with deep Q-network for industry 4.0 automation and orchestration. *IEEE Trans Ind Inform* 17(4):2852–2859. <https://doi.org/10.1109/TII.2020.3000502>
32. Pokhrel SR, Pan L, Kumar N, Doss R, Vu HL (2021) Multipath TCP meets transfer learning: a novel edge-based learning for industrial IoT. *IEEE Internet Things J* 8(13):10299–10307. <https://doi.org/10.1109/JIOT.2021.3056466>
33. Cao Y, Ji R, Ji L, Lei G, Wang H, Shao X, (Early Access, (2022)  $l^2$ -MPTCP: a learning-driven latency-aware multipath transport



- scheme for industrial internet applications. *IEEE Transactions on Industrial Informatics*. <https://doi.org/10.1109/TII.2022.3151093>
34. Kanagarathinam MR, Natarajan H, Arunachalam K, Sandeep I, Sunil V (2020) SMS: Smart multipath switch for improving the throughput of multipath TCP for smartphones. In: *Proceedings of 2020 IEEE Wireless Communications and Networking Conference (WCNC)*, Seoul, Korea (South), pp 1–6. <https://doi.org/10.1109/WCNC45663.2020.9120463>
  35. Xu C, Qin J, Zhang P, Gao K, Grieco LA, (2021) Reinforcement learning-based Mobile AR/VR Multipath Transmission with Streaming Power Spectrum Density Analysis. *IEEE Transactions on Mobile Computing*. <https://doi.org/10.1109/TMC.2021.3082912>
  36. Padhye J, Firoiu V, Towsley D, Kurose J (1998) Modeling TCP throughput: a simple model and its empirical validation. In: *Proceedings of the ACM SIGCOMM'98 Conference on Applications, Technologies, Architectures, and Protocols for Computer Communication (SIGCOMM)*, Vancouver, British Columbia, Canada, pp 303–314. <https://doi.org/10.1145/285237.285291>
  37. Dong P, Yang W, Tang W, Huang J, Wang H, Pan Y, Wang J (2018) Reducing transport latency for short flows with multipath TCP. *J Netw Comput Appl* 108:20–36. <https://doi.org/10.1016/j.jnca.2018.02.005>
  38. Sargent M, Chu J, Paxson DV, Allman M (2011) Computing TCP's retransmission timer. <https://datatracker.ietf.org/doc/rfc6298/>

**Publisher's Note** Springer Nature remains neutral with regard to jurisdictional claims in published maps and institutional affiliations.

Springer Nature or its licensor (e.g. a society or other partner) holds exclusive rights to this article under a publishing agreement with the author(s) or other rightsholder(s); author self-archiving of the accepted manuscript version of this article is solely governed by the terms of such publishing agreement and applicable law.



**Du Chen** received the B.S. degree in information engineering from Beijing Technology and Business University, Beijing, China, in 2017. He is currently pursuing the Ph.D. degree with the School of Electronic and Information Engineering, Beijing Jiaotong University, Beijing, China. He has participated in several programs, including the National Key Research and Development Program of China and the National Natural Science Foundation of China. His research interests include future Internet, multipath TCP, and programmable data plane.



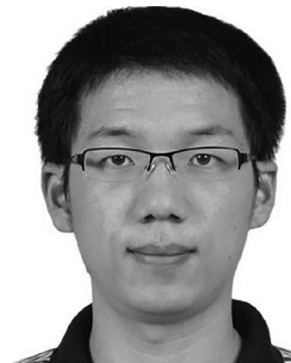
**Deyun Gao** received the B.S. and M.S. degrees in electrical engineering and the Ph.D. degree in computer science from Tianjin University, Tianjin, China, in 1994, 1999, and 2002, respectively. He spent one year as a Research Associate with the Department of Electrical and Electronic Engineering, Hong Kong University of Science and Technology, Hong Kong. Then, he spent three years as a Research Fellow with the School of Computer

Engineering, Nanyang Technological University, Singapore. In 2014, he was a Visiting Scholar with the University of California at Berkeley, Berkeley, CA, USA. He is currently a Full Professor with Beijing Jiaotong University, Beijing, China. His research interests include Internet of Things, vehicular networks, software defined networking, and programmable data plane.



tipath TCP, and programmable data plane.

**Lu Jin** received the B.S. and M.S. degrees in communication and information systems from Beijing Jiaotong University, Beijing, China, in 2018 and 2021, respectively. She has participated in several programs, including the National Key Research and Development Program of China and the National Natural Science Foundation of China. Currently, she is a software engineer in the Agricultural Bank of China. Her research interests include software defined networking,



including *IEEE Communications Magazine*, *IEEE Wireless Communications*, *IEEE Network*, *IEEE Transactions on Vehicular Technology*, *IEEE Communications Letters*, *IFIP Networking*, *IEEE ICC*, and *IEEE GLOBECOM*. His research interests include key technologies for network analytics, future Internet, 6G networks, and vehicular networks.

**Wei Quan** received the Ph.D. degree in communication and information system from Beijing University of Posts and Telecommunications, Beijing, China, in 2014. He spent one year as a Visiting Scholar with Department of Electrical and Computer Engineering, University of Waterloo, Canada. He is currently a Full Professor with Beijing Jiaotong University, Beijing, China. He has authored or coauthored more than 50 papers in prestigious international journals and conferences,



and has also served on the editorial board of several international journals. His research has resulted in many papers, books, patents, systems, and equipment, in the areas of communications and computer networks.

**Hongke Zhang** received the M.S. and Ph.D. degrees in electrical and communication systems from the University of Electronic Science and Technology of China, Chengdu, China, in 1988 and 1992, respectively. From 1992 to 1994, he was a Postdoctoral with Beijing Jiaotong University, Beijing, China. He is currently a Full Professor with Beijing Jiaotong University, where he directs the National Engineering Research Center of Advanced Network Technologies. He is the Chief Scientist of a National Basic Research Program of China (973 Program) and