IEEE Access
Multidisciplinary | Rapid Review | Open Access Journal

SPECIAL SECTION ON ROADMAP TO 5G: RISING TO THE CHALLENGE

Received January 3, 2019, accepted January 24, 2019, date of publication February 7, 2019, date of current version March 20, 2019.

Digital Object Identifier 10.1109/ACCESS.2019.2898110

Performance Evaluation of Multipath TCP Scheduling Algorithms

PINGPING DONG[1], JINGYUN XIE[1], WENSHENG TANG[1], NAIXUE XIONG[2], HUA ZHONG[1], AND ATHANASIOS V. VASILAKOS[3]

[1]Hunnan Provincial Key Laboratory of Intelligent Computing and Language Information Processing, Hunan Normal University, Changsha 410081, China
[2]College of Intelligence and Computing, Tianjin University, Tianjin 300350, China
[3]Department of Computer Science, Electrical and Space Engineering, Lulea University of Technology, 93187 Skellefteå, Sweden

Corresponding author: Naixue Xiong (xiongnaixue@gmail.com)

This work was supported in part by the National Natural Science Foundation of China under Grant 61602171, in part by the Hunan Provincial Science and Technology Project Foundation under Grant 2018TP1018, and in part by the Scientific Research Fund of Hunan Provincial Education Department under Grant 17C0960.

ABSTRACT One of the goals of 5G is to provide enhanced mobile broadband and enable low latency in some use cases. To achieve this aim, the Internet Engineering Task Force has proposed the Multipath TCP by utilizing the feature of dual connectivity in 5G, where a 5G device can be served by two different base stations. However, the path heterogeneity between the 5G device and the server may cause a packet out-of-order problem. The researchers proposed a number of scheduling algorithms to tackle this issue. This paper introduces the existing algorithms, with the aim to make a thorough comparison between the existing scheduling algorithms and provide the guidelines for designing new scheduling algorithms in 5G, we have conducted an extensive set of emulation studies based on the real Linux experimental platform. The evaluation covers a wide range of network scenarios to investigate the impact of different network metrics, namely, RTT, buffer size, and file size on the performance of existing widely deployed scheduling algorithms.

INDEX TERMS 5G, enhanced broadband, Multipath TCP, out-of-order, scheduling algorithm.

I. INTRODUCTION

Once a decade, a new generation of mobile network technology comes along, starting in the 1980s, where the first mobile networks appeared, until 2018, where the first 5G standard is completed. 5G is seeking to achieve some key goals like enhanced broadband [1], [2] and ultra-low latency and MPTCP can be closely integrated to with the 5G stack [3], [4] to achieve this aim with regards to dual connectivity.The key motivation behind MPTCP is the trend toward providing trusted and reliable connectivity in the future Internet. The research in this context focuses on faster downloads, lower data transfer costs and seamless switching between different interfaces, particularly the wireless ones such as Wi-Fi and cellular networks [5], [6].

One of the main reasons for sub-optimal performance of MPTCP, particularly in terms of aggregate capacity, is that of packet reordering caused by path heterogeneity. More specifically, the packet with the lower sequence number arrives at the receiver later than the packet with the higher sequence number. The receiver has to buffer the packets with the lower sequence number. Until it receives all the packets whose sequence number is lower than the higher sequence number, the data can be submitted to the upper layer [7]. Out-of-order also can cause end-to-end delay and application throughput are reduced. To solve these problems, researchers proposed many algorithms, which can be classified into two aspects, namely, congestion control and path scheduling.

The congestion control algorithm of MPTCP can adaptively adjust the transmission rate of each subflow, and it attempt to shift traffic from more congested path to a less congested path, thereby improving throughput and link utilization. So far, researchers have proposed many MPTCP congestion control algorithms, like LIA (Linked Increases Algorithm) [8], [9], SEMICOUPLED [9], OLIA (Opportunistic Linked-Increases Algorithm) [10], Balia (Balanced Linked Adaptation) [11], wVegas (Weighted Vegas) [12], mVeno [13], EWTCP (equally-weighted TCP) [14], COUPLED [15], [16], RTT-Compensator [17], TCP Vegas [18], MPVeno [19], MPTCPPW [20].

The associate editor coordinating the review of this manuscript and approving it for publication was Jordi Mongay Batalla.

Congestion control solves the problem of MPTCP from controlling the transmission rate of each subflow, and the scheduling algorithm is designed to distribute data packets on multiple paths based on each path's congestion window size controlled by the congestion control algorithm. The path scheduling algorithm is crucial for us to study MPTCP [21]. There are many existing scheduling algorithms, Like, Round-Robin (RR) [22], Constraint-based proactive scheduling (CP) [23], Highest Sending Rate (HSR) [24], Largest Window Space (LWS) [24], Lowest Time/Space (LTS) [24], Fine-grained forward Prediction based Dynamic Packet Scheduling (F^2P-DPS) [25]. In this paper, we analyzed four widely-deployed scheduling algorithms, namely, LowRTT, OTIAS, DAPS and BLEST. The Lowest-RTT-First (LowRTT) [26] first sents the packet on the subflow with the lowest RTT, until its congestion window is filled with packet. Then, the packet is sent on the subflow with the next higher RTT. The Delay-Aware Packet Scheduler (DAPS) [21], [27] determines the amount of data that should be sent in the next round of each path according to the ratio of the forward transmission delay and the size of the congestion window (CWND), so that the data packets arrive in order. The Out-of-order Transmission for In Order Arrival Scheduler (OTIAS) [28], when transmitting a new data packet, estimates the time at which the data packet arrives at the receiver, and selects the subflow with the earliest arrival time to transmit data to arrive in order. The Blocking Estimation-based MPTCP Scheduler (BLEST) [29] predicts whether the subflow will occur head-of-line blocking, and then estimates how much data each subflow should be send, and tries to make the data send on the fast subflow, even if the slow subflow has space on the congestion window, and tries to avoid sending on the slow subflow to avoid buffer blocking.

Based on the real experimental platform of Linux, we conduct extensive experiments to investigate the performance of each algorithm, concludes the aspects that influence the performance of the scheduling algorithms, and further provides guidelines for designing new scheduling algorithms.

The rest of the paper is organized as follows. The describe of MPTCP background is presented in Section 2. Section 3 analyzes the MPTCP scheduling algorithms. We evaluate MPTCP algorithms with the Linux testbed in Section 4. Finally, Section 5 concludes the paper.

II. MPTCP BACKGROUND

As an extension of the TCP protocol, MPTCP supports simultaneous transmission of data through multiple paths, i.e. a protocol in which MPTCP can simultaneously use multiple network interfaces. As shown in Fig.1 [30], the MPTCP is located between the application layer and the network layer, and can be further divided into an MPTCP layer and a TCP sub-flow layer. The MPTCP layer is transparent to the application layer. The TCP subflow layer provides multiple paths to the application.

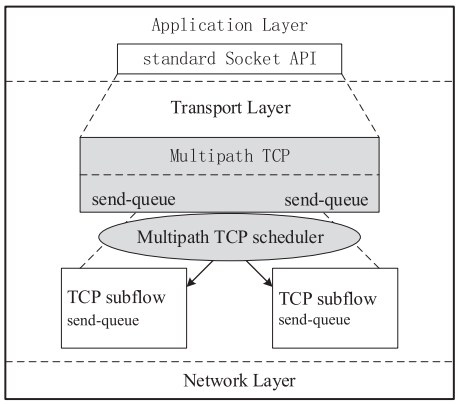

FIGURE 1. The protocol stack of MPTCP.

Multipath TCP is a set of TCP extensions defined in RFC 6824, and allows a single TCP connection to send and receive data simultaneously using different IP addresses [31]. The life cycle of an MPTCP connection includes three phases, namely initial connection, data transfer, and closing connection.

In the initial connection phase, the difference between MPTCP and TCP is that MPTCP has a four-way handshake before the multipath is enabled. MPTCP and TCP have similar three-way handshake. But the SYN, SYN/ACK and ACK packets have the MPTCP_CAPABLE option. After the TCP connection has been established, the client can advertise other addresses by sending a TCP segment with the ADD_ADDR option.

During the MPTCP data transfer phase, the subflows are linked together by a single multipath TCP connection, and both can transmit data. To ensure the reliability of the orderly transfer of data on subflow, MPTCP uses two principles. First, each subflow is equivalent to a regular TCP connection, and with its own 32-bit sequence number space. Second, MPTCP maintains a 64-bit data sequence number space. The DSN_MAP and DSN_ACK options use these data sequence numbers.

In the closed connection phase, when the sender informs the receiver that no data will be sent, the Data FIN option will be part of the data sequence signal. Data FIN of MPTCP has the same semantics and behavior as a regular TCP FIN, but it belongs to the connection level. After successfully receiving all the data on the MPTCP connection, this message is acknowledged at the connection level using DATA_ACK. After DATA_ACKs confirms the DATA_FIN of both hosts, the connection is considered closed. And both hosts should send FINs on all subflows.

The packet scheduler gets the data from the application layer and then distributes the data to each subflow. The main task of data scheduling is to distribute the data to each sub-flow reasonably, so that the data arrives at the receiver as much as possible in order to reduce the head-of-line blocking at the receiving end. The scheduler how to uses a suitable scheduling algorithm distributes the data to each subflow is very important.

III. ANALYSIS OF MPTCP SCHEDULING ALGORITHMS

Due to TCP cannot use multiple interfaces, we use MPTCP to transmit data. However, in the research of MPTCP, there are still problems such as out-of-order. In order to solve these problems, researchers have done a lot of work and proposed many scheduling algorithms. But, the different scheduling algorithms use different design principles. In this section, we first give an overview of existing algorithms, and then detailed analyze the four widely-deployed algorithms, namely, LowRTT, DAPS, OTIAS and BLEST.

A. EXISTING SCHEDULING ALGORITHMS

Researchers have proposed a number of scheduling algorithms. The RR algorithm [22] is a simple scheduling mechanism. There is no priority between subflows in the RR algorithm, which selects subflows in a round-robin fashion. The LowRTT [26] first sents the packet on the subflow with the lowest RTT, until its congestion window is filled with packet. Then, the packet is sent on the subflow with the next higher RTT. The DAPS [21], [27] determines the amount of data that should be sent in the next round of each path according to the ratio of the forward transmission delay and the size of the congestion window (CWND), so that the data packets arrive in order. The OTIAS [28] algorithm, when transmitting a new data packet, estimates the time at which the data packet arrives at the receiver, and selects the subflow with the earliest arrival time to transmit data. The BLEST [29] predicts whether the subflow will occur head-of-line blocking, and then estimates how much data each subflow should be send, and tries to make the data send on the fast subflow, even if the slow subflow has space on the congestion window, and tries to avoid sending on the slow subflow to avoid buffer blocking. The CP [23] estimates the out-of-order packets based on the performance difference between the subflows, and compares the estimated out-of-order packets and buffer size to assign packets to subflows. In addition, network delay constraints are used to adjust the trade-off between throughput and delay performance. The HSR [24] measures the path's MSS and instantaneous RTT on the path r, uses the product of the ratio of the two and the current congestion window to calculate the transmission rate, and selects the path with a maximum sending rate to transmit data. The LWS [24] uses the difference between the current window w_r and the current amount of in-flight packets f_r as the window space w_s of the path r, and uses a path with larger w_s to transmit data. When the path is in the lost recovery state, which may be w_r shorter than or equal to f_r, resulting w_s in a negative value, the LowRTT scheduling mechanism is used. The LTS [24] selects a path with lower latency and maximum window space to transmit data based on the lowest ratio between the current $sRTT$ and window space w_s. As in LWS, when w_s is negative or null, the LowRTT scheduling mechanism is used. These algorithms all use different design methods. The F^{2}P-DPS [25] considers the TCP characteristics of the subflow and the packet loss rate of the path. The smoothed RTT obtained by the subflow and the packet loss rate are used to estimate the amount of data that the subflow may send in the future.

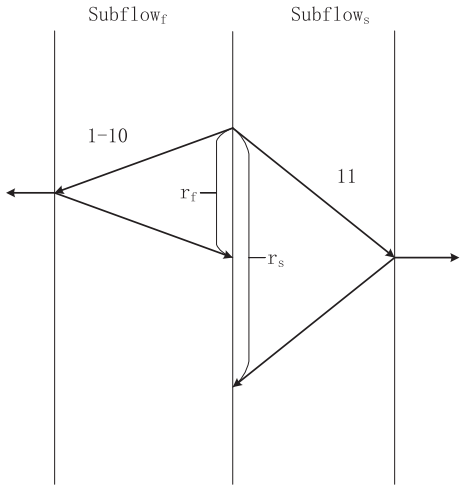

FIGURE 2. Two subflows packet scheduling diagrams in LowRTT algorithm.

B. LOWEST-RTT-FIRST (LowRTT)

The LowRTT algorithm is the default scheduler, which first sends the packet on the subflow with the lowest RTT, until its congestion window is filled with packet. Then, the packet is sent on the subflow with the next higher RTT. The LowRTT algorithm makes the sub-flow with good path quality bear more data and has a certain load balancing effect. This is better than the RR algorithm, but neither algorithm considers the packet ordering. When the path difference is large, the path utilization will decrease.

As shown in Fig.2, there are two subflows ($subflow_f$ and $subflow_s$). The congestion window of two subflows are 10, and RTT of two subflows is $r_f = 10ms$, $r_s = 20ms$, i.e., the time required to send 2 rounds of data over $subflow_f$ is the same as the time required to send 1 round of packets over $subflow_s$. When there are 11 packets to be sent, the $subflow_f$ takes $1, 2, 3, \ldots, 10$ Packets, $subflow_s$ take 11 packets. The completion time of the $subflow_f$ is r_f, the $subflow_s$ completion time is r_s, $r_s > r_f$. Thus, it can be seen that the entire completion time is slowed down by the slow flow. When the data is relatively large, the data packets cannot arrive at the receiver in order.

C. DELAY-AWARE PACKET SCHEDULER (DAPS)

The DAPS algorithm aims to reduce the blocking time of the receive buffer. In the existing network, the link is asymmetric (i.e., each path has different delays, different capacity, different congestion windows), so the packets that can be scheduled for each path is different. The DAPS algorithm considers the difference of the path. It uses the ratio of the forward transmission delay ($RTT/2$) and the size of the congestion window (CWND) to determine the amount of data that each

path should send next time, so that the packets arrive in order, and reducing Head-Of-Line Blocking.

In the DAPS algorithm, it is assumed that the data on an MPTCP connection needs to be scheduled to two TCP subflows ($subflow_f$ and $subflow_s$). The congestion window sizes of the two subflows is respectively $cwnd_f$ and $cwnd_s$ the round-trip time is respectively r_f and r_s. If the ratio of the forward transmission delay is equal to the ratio of the Round-Trip Time, the ratio of the forward transmission delay of the two subflows can be expressed as $\eta = \left\lfloor \frac{r_s}{r_f} \right\rfloor$. When η less than $cwnd_f$, in the next round of data transmission, the data sent by the $subflow_f$ is $TSN1, \ldots, TSN1+\eta$, the data sent by the $subflow_s$ is $TSN1+\eta+1, \ldots, TSN1+\eta+cwnd_s$. When η greater than $cwnd_f$, in the next round of data transmission, the data sent by the $subflow_f$ is $TSN1, \ldots, TSN1 + cwnd_f$, The data sent by the $subflow_s$ is $TSN1 + cwnd_f + 1, \ldots, TSN1 + cwnd_f + cwnd_s$.The algorithm is as follows:

Algorithm 1 Delay-Aware generateSchedule for Two Paths

Require: $\eta_{pkts} > cwnd_f - unack_f$ {Too many packets for the fast path only}

Ensure: S is a packet/path schedule so packets are received in order

1: $S \leftarrow \phi$
2: $\eta = \left\lfloor \frac{r_s}{r_f} \right\rfloor$ {Note: $r_s \geq r_f$}
3: $max_f \leftarrow min(\eta, cwnd_f - unack_f)$ {Maximum number of packets to send on the fast path}
4: **for** $j = 1, \ldots, max_f$ **do** {Schedule for the fast path}
5: $s_j \leftarrow (getNextUnsentChunk, p_f)$ {j^{th} chunk}
6: Append s_j to S
7: **end for**
8: **for** $j = 1, \ldots, cwnd_s - unack_s$ **do**{Schedule for the slow path}
9: $s_j \leftarrow (getNextUnsentChunk, p_s)$ {$(\max_f + j)^{th}$ chunk}
10: Append s_j to S
11: **end for**
12: **return** S

The DAPS algorithm determines the scheduling of N data packets based on the given time information, and ensures that the data packets can arrive at the receiver in order, which is better than the LowRTT algorithm. But, the algorithm pre-allocates data packets for each path. When the link characteristics change, a large number of data packets are accumulated on the link, and the data needs to be re-scheduled for one cycle, which is insensitive to link changes [32].

D. OUT-OF-ORDER TRANSMISSION FOR IN ORDER ARRIVAL SCHEDULER (OTIAS)

The OTIAS algorithm mitigates jitter by sending packets out of order on different subflows, enabling packets to arrive in order at the receiver, and hoping to schedule more segments on the subflow than it currently. The OTIAS scheduling algorithm is based on data scheduling for each packet, i.e., for each data packet, the one-way transmission delay of the data packet to the receiver of each path is estimated, and the data packet is sequentially scheduled to the path with a small forward transmission delay. If there is space in the CWND, the segment will be sent immediately. If the CWND is full, the segment must wait in the queue of the subflow. We estimate the one-way transmission delay for the packet i to be transmitted in the subflow j as follows:

$$\begin{aligned} pkt\_can\_be\_sent\_j &= cwnd_j - unacked_j \\ RTT\_to\_wait_i^j &= \left\lfloor \frac{no\_yet\_sent\_j - pkt\_can\_be\_sent\_j}{cwnd_j} \right\rfloor \\ T_i^j &= (RTT\_to\_wait_i^j + 0.5) \times srtt_j. \end{aligned}$$

In the above formula, $pkt\_can\_be\_sent\_j$ is indicated a data packet that can be immediately transmitted in a subflow, $no\_yet\_sent\_j$ is indicated a data packet that has not been sent in the subflow j, and $RTT\_to\_wait_i^j$ is a waiting time for the packet i to be transmitted on the subflow j.

Compared with the DAPS algorithm, the OTIAS algorithm adjusts the scheduling when the packets are dequeued, while the OTIAS adjusts the scheduling when packets are enqueued. The OTIAS can respond to network changes more dynamically than the DAPS. However, OTIAS also has shortcomings. When the difference of two paths is large and there are packet loss in the link, a large number of data packets will be accumulated on the low-latency link, and the Hol-blocking problem cannot be solved well.

E. BLOCKING ESTIMATION-BASED MPTCP SCHEDULER (BLEST)

The BLEST algorithm dynamically adapts the schedule by estimating whether a head-of-line blocking will occur, reducing head-of-line blocking, false retransmissions, and improving application performance in heterogeneous scenarios. BLEST estimates packets that can be transmitted in a fast subflow without Hol-blocking. Because the RTT of the slow subflow is relatively large, the data packet arrives at the receiver relatively late. When the amount of data is relatively large, a large number of out-of-order packets are generated, thereby increasing the completion time. Therefore, BLEST tries to use fast subflows to send packets so that they can arrive in order.

Suppose there are two subflows ($subflow_f$ and $subflow_s$) that can send data with round trip times of RTT_f and RTT_s. If the data is sent on a slow $subflow_s$, BLEST assumes that one segment will occupy space at least RTT_s in the MPTCP's send window ($MPTCP_{sw}$). In order to send data through the fast $subflow_f$ as much as possible, it is estimated that the packet X that can be sent on the fast $subflow_f$ without the Hol-blocking during the period RTT_s as follows:

$$\begin{aligned} rtts &= RTT_s/RTT_f \\ X &= MSS_f \cdot (CWND + (rtts - 1)/2) \cdot rtts \end{aligned}$$

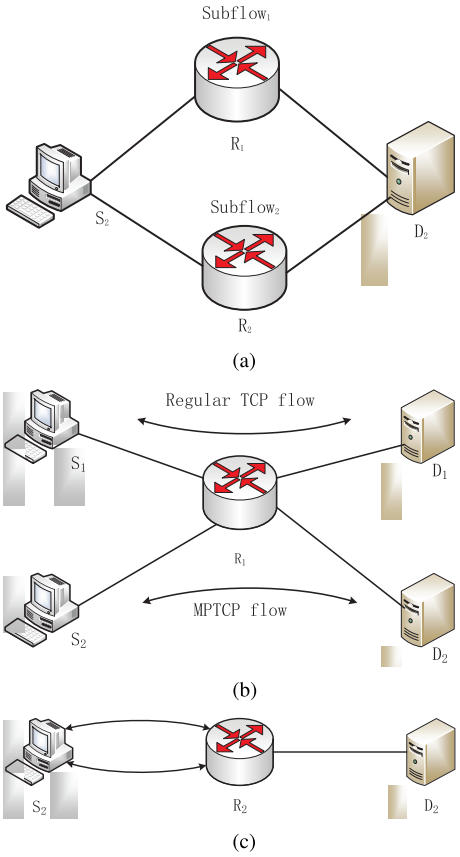

FIGURE 3. TestBed topology. (a) Non-shared network topology. (b) Competitive network topology. (c) Shared network topology.

The X estimation may be inaccurate and will be constrained by a λ value, which is described in [29]. If $x \times \lambda > |M| - MSS_s \cdot (inflight + 1)$, i.e. the next segment will not be sent on $subflow_s$. Instead, the scheduler waits for the faster subflow to become available. The BLEST algorithm can skip a subflow and wait for a more favorable subflow to send data, which can reduce the risk of Hol-blocking and thus the number of retransmissions triggered. BLEST outperforms the other three algorithms. However, BLEST does not consider idle fast subflow due to nothing to send [33], it can't efficiently utilize the faster paths.

IV. EVALUATION

In this section, we evaluate the performance of the four algorithms based on the real Linux experimental platform with the topology shown in Fig.3 under different RTT, buffer size, and file size to investigate the performance of each algorithm under different network parameters and different topologies. We provide guidance for designing new scheduling algorithms and provide convenience for future work.

A. EXPERIMENTAL PLATFORM

The deployed experiment testbed consists of two file servers, two computers with WANem, and two clients, which constitutes the network topology show in Fig.3 by means of routing configurations. The topology is widely used in existing works [27], [34]. Both the clients and the servers are running the Linux ubuntu 12.10 operating system with the kernel version 3.14.33 that has already applied protocol patches. The servers are running on the Dell T1500, equipped with the Intel Xeon E5620 (2.4 GHz/12M), 16 GB RAM and a 600 GB Hard Disk. The clients are running on the DELL optiplex 745, equipped with Intel PentiumD 3.4G processor, 512 MB RAM and a 160 GB Hard Disk. As shown in Fig.3, one of the servers labeled $S2$ is equipped with two Gigabit network interface cards to establish two subflows between the MPTCP client $D2$. We consider this as the common scenario (e.g., a client having two access networks like WiFi/4G) [34], [35]. As shown in Fig.3, $R1$ and $R2$ serve as two routers which run WANem to construct a two-way bottleneck link. WANem is a wide area network emulator that supports various wide area network features such as bandwidth limitation, latency, packet loss, network disconnection and so on.

In the configuration, the commands *sysctl net.mptcp.mptcp_enabled* and *sysctl net.mptcp.mptcp_path_manager* are utilized to enable MPTCP and select MPTCP path management. We also use these commands *sysctl net.ipv4.tcp_congestion_control* and *sysctl net.mptcp.mptcp_scheduler* to configure the congestion control and scheduling algorithms. In the experiments, olia, the default congestion control algorithm is used.

We use the GNU Wget to retrieve binary document over HTTP to generate TCP data traffic. The Binary files range from 16 KB to 8 MB, the round-trip time ranges from 20 ms to 400 ms, and the bottleneck bandwidth varies from 2 Mbps to 100 Mbps.

B. PERFORMANCE EVALUATION IN A NON-SHARED SCENARIO

Fig.3(a) shows the non-shared scenario. There are no shared bottlenecks between the MPTCP's two subflows. Since the four algorithms have different processing methods in path heterogeneity, we setting different RTT values in the experiment. In addition, different buffer size and file size have different effects on the four algorithms, so we also set different buffer size values and different file size values.

1) PATH HETEROGENEITY TEST

In this path heterogeneity test, the bandwidth is 100 Mbps and the number of concurrent flows is 20, RTT_1 is 10 ms, and RTT_2 ranges from 10ms to 600ms. The number of timeouts and the flow completion time are shown in Fig.4(a) and Fig.4(b), respectively. As depicted in Fig.4(a), the LowRTT has largest number of timeout, and the BLEST has least number of timeout. As a result, BLEST performs best, followed by OTIAS and DAPS, and LowRTT performs worst as described in Fig.4(b). The reason lies in that although all the four algorithms consider path heterogeneity, LowRTT does not consider the effects of packets out-of-order. BLEST tries to use fast subflows to send packets so that they can arrive in order, and can reduce head-of-line blocking.

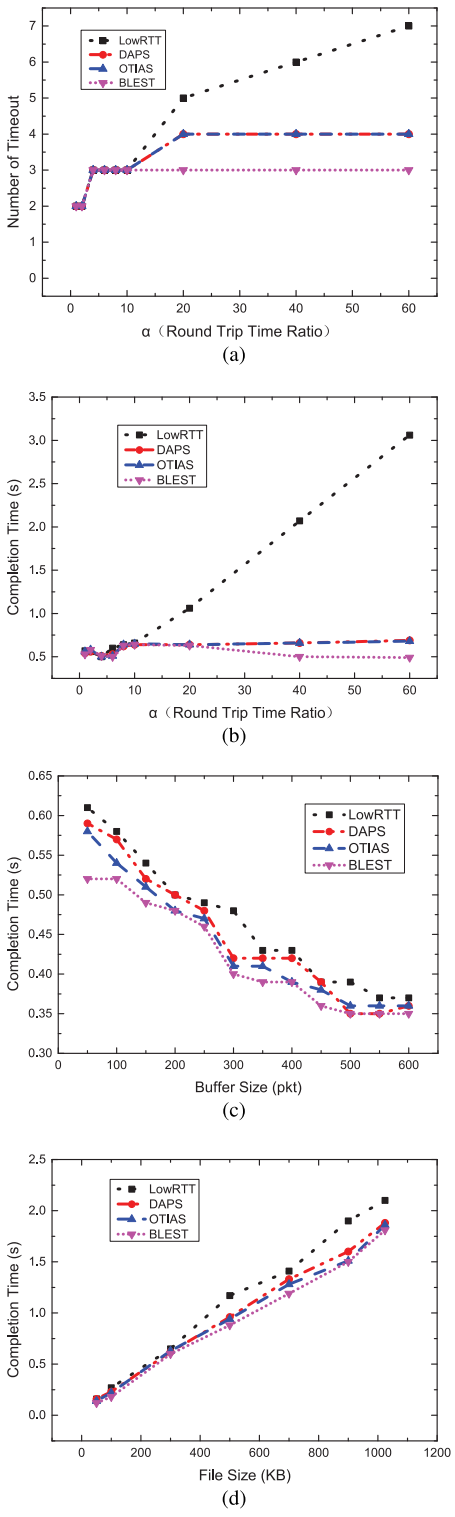

FIGURE 4. Experimental results in the non-shared scenario. (a) Number of timeout with different α. (b) Completion time with different α. (c) Completion time with different receive buffer sizes. (d) Completion time with different file sizes.

2) DIFFERENT BUFFER SIZE

In these experiments, the bandwidth is 5 Mbps, the number of concurrent flows is 20, the *RTT* of each path is 20 ms, and the file size is 300 KB. Fig.4(c) shows the experimental

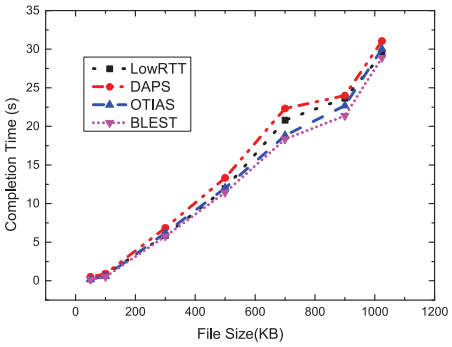

FIGURE 5. Completion time with different file sizes in TCP.

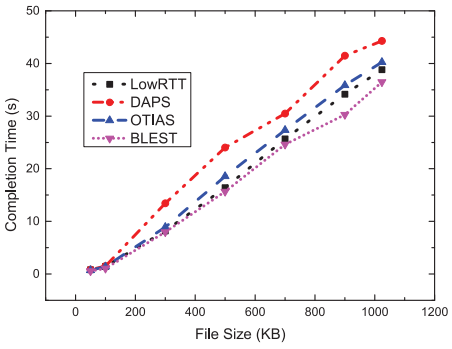

FIGURE 6. Completion time with different file sizes in MPTCP.

results, where BLEST outperforms other algorithms, followed by OTIAS and DAPS. This is because that BLEST can dynamically estimate whether a head-of-line blocking will occur, and reduce the number of out-of-order packets. Although DAPS and OTIAS have the same goal to reduce Hol-blocking, DAPS is insensitive to link changes. When the link characteristics change, a large number of data packets are accumulated on the link, and the data needs to be re-scheduled for one cycle. In addition, OTIAS can respond to network changes more dynamically than the DAPS. However, when the difference of two paths is large, a large number of data packets will be accumulated on the low-latency link, and the Hol-blocking problem cannot be solved well.

3) DIFFERENT FILE SIZE TESTS

In different file size tests, the bandwidth is 5 Mbps, the number of concurrent flows is 20, the *RTT* of each path is 20 ms and the buffer size is 20 packets. The results are shown in Fig.4(d). The results are consistent with the results when the buffer size is different. Specifically, BLEST performs best, OTIAS outperforms DAPS, and LowRTT performs worst. The reason is the same as different buffer size test and BLEST outperforms the other algorithms.

C. PERFORMANCE EVALUATION UNDER COMPETING TRAFFIC SCENARIOS

In the competing traffic test network scenario, the regular TCP traffic competing the same bottleneck with MPTCP traffic as shown in Fig.3(b). In these experiments,

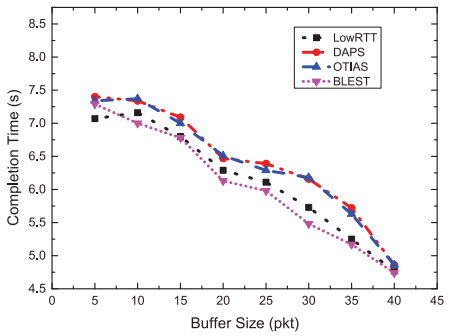

FIGURE 7. Completion time with different receive buffer sizes.

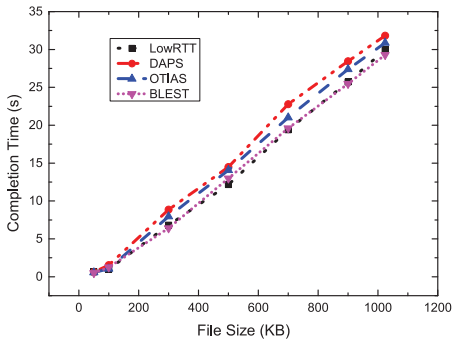

FIGURE 8. Completion time with different file sizes.

the bandwidth of MPTCP and TCP are 5Mbps, the number of concurrent flows is 20, the RTT of each path is 20ms, and the buffer size is 20 packets. The experimental results are shown in Fig.5 and Fig.6. According to the two figures, the completion time becomes larger with the increasing file size. In addition, the difference between the flow completion time of TCP and multipath algorithms is largest with BLEST and is smallest with LowRTT. This further validates that BLEST also outperforms other algorithms in the competing network scenario.

D. PERFORMANCE EVALUATION UNDER SHARED NETWORK SCENARIOS

In the shared test network, the clients access link (downlink) is the bottleneck and it is therefore shared among the two flows as shown in Fig.3(c). We evaluate the performance of the four algorithms on the bottleneck link by setting different file size and buffer size, where the bandwidth is 5 Mbps, the number of concurrent flows is 20 and the RTT of each path is 20ms.

1) DIFFERENT BUFFER SIZE

In these experiments, the file size is 300 KB. The completion time, shown in Fig.7, becomes smaller with the increasing buffer size. The BLEST performs best, followed by LowRTT. The DAPS performs worse compared to OTIAS. This is because that BLEST is able to react more dynamically to network changes, followed by LowRTT and OTIAS. DAPS is insensitive to link changes.

2) DIFFERENT FILE SIZE

Fig.8 shows the completion time becomes larger with the increasing file size when the buffer size is 20 packets. The results are in accordance with those when the buffer size is different. As analyzed above, the reason lies in that BLEST can respond to network changes more dynamically than the LowRTT and OTIAS, and DAPS is insensitive to link changes.

V. CONCLUSIONS

MPTCP can be closely integrated to with the 5G stack to improve 5G network capacity. This paper firstly introduce the existing MPTCP scheduling algorithms and summarizes their characteristics. Then, based on the real Linux experimental platform, we evaluate the performance of four widely-deployed scheduling algorithms under different network parameters and different topologies. In MPTCP, path heterogeneity is an important factor that affecting network performance. When the path difference is large, we should pay attention to the processing of out-of-order packets, so that the packets can arrive at the receiving end in order. In future work, we plan to study the performance of the algorithm in the video stream scene and fully exploit the potential of MPTCP.

REFERENCES

[1] G. Liu *et al.*, "3-D-MIMO with massive antennas paves the Way to 5G enhanced mobile broadband: From system design to field trials," *IEEE J. Sel. Areas Commun.*, vol. 35, no. 6, pp. 1222–1233, Jun. 2017.
[2] M. Hashemi, C. E. Koksal, and N. B. Shroff, "Out-of-band millimeter wave beamforming and communications to achieve low latency and high energy efficiency in 5G systems," *IEEE Trans. Commun.*, vol. 66, no. 2, pp. 875–888, Feb. 2018.
[3] U. Chunduri *et al.*, *Considerations for MPTCP Operation in 5G*, document draft-defoy-mptcp-considerations-for-5g-00, 2007.
[4] C. Lee, S. Song, H. Cho, G. Lim, and J.-M. Chung, "Optimal multipath TCP offloading over 5G NR and LTE networks," *IEEE Wireless Commun. Lett.*, to be published.
[5] J. Zeng, Y. Cao, F. Ke, M. Huang, G. Zhang, and W. Lu, "Performance evaluation of secure multipath retransmission mechanism in next generation heterogeneous communication systems," *IET Netw.*, vol. 7, no. 2, pp. 61–67, 2018.
[6] K. Xue *et al.*, "DPSAF: Forward prediction based dynamic packet scheduling and adjusting with feedback for multipath TCP in lossy heterogeneous networks," *IEEE Trans. Veh. Technol.*, vol. 67, no. 2, pp. 1521–1534, Feb. 2018.
[7] H. Huang, "Weight-based data scheduling algorithm designing for MPTCP," *Comput. Eng. Softw.*, vol. 37, no. 2, pp. 77–80, Feb. 2016.
[8] C. Raiciu, M. Handley, and D. Wischik, *Coupled Congestion Control for Multipath Transport Protocols*, document RFC 6356, 2011.
[9] D. Wischik, C. Raiciu, A. Greenhalgh, and M. Handley, "Design, implementation and evaluation of congestion control for multipath TCP," in *Proc. Usenix Conf. Netw. Syst. Design Implement.*, 2011, pp. 99–112.
[10] R. Khalili, N. Gast, M. Popovic, and J.-Y. Le Boudec, "MPTCP is not Pareto-optimal: Performance issues and a possible solution," *IEEE/ACM Trans. Netw.*, vol. 21, no. 5, pp. 1651–1665, Oct. 2013.
[11] Q. Peng, A. Walid, J. Hwang, and S. H. Low, "Multipath TCP: Analysis, design, and implementation," *IEEE/ACM Trans. Netw.*, vol. 24, no. 1, pp. 596–609, Feb. 2016.
[12] Y. Cao, M. Xu, and X. Fu, "Delay-based congestion control for multipath TCP," in *Proc. IEEE Int. Conf. Netw. Protocols*, Oct./Nov. 2012, pp. 1–10.
[13] P. Dong, J. Wang, J. Huang, H. Wang, and G. Min, "Performance enhancement of multipath TCP for wireless communications with multiple radio interfaces," *IEEE Trans. Commun.*, vol. 64, no. 8, pp. 3456–3466, Aug. 2016.
[14] M. Honda, Y. Nishida, L. Eggert, P. Sarolahti, and H. Tokuda, "Multipath congestion control for shared bottleneck," in *Proc. PFLDNeT Workshop*, vol. 357, 2009, p. 378.

[15] H. Han, S. Shakkottai, C. V. Hollot, R. Srikant, and D. Towsley, "Overlay TCP for multi-path routing and congestion control," in *Proc. IMA Workshop Meas. Modeling Internet*, 2004, pp. 1–24.
[16] S. Jiang, "Granular differentiated queueing services for QoS: Structure and cost model," *ACM SIGCOMM Comput. Commun. Rev.*, vol. 35, no. 2, pp. 13–22, 2005.
[17] C. Raiciu, D. Wischik, and M. Handley, "Practical congestion control for multipath transport protocols," Dept. Comput. Sci., Univ. College London, London, U.K., Tech. Rep, 2009. [Online]. Available: http://nrg.cs.ucl.ac.uk/mptcp/mptcp-techreport.pdf
[18] L. S. Brakmo, S. W. O'Malley, and L. L. Peterson, "TCP VEGAS: New techniques for congestion detection and avoidance," *SIGCOMM*, vol. 24, no. 4, pp. 24–35, 1994.
[19] T.-A. Le, "Improving the performance of multipath congestion control over wireless networks," in *Proc. Int. Conf. Adv. Technol. Commun.*, 2013, pp. 60–65.
[20] T. A. Le, C. S. Hong, and E.-N. Huh, "Coordinated TCP westwood congestion control for multiple paths over wireless networks," in *Proc. Int. Conf. Inf. Netw.*, 2012, pp. 92–96.
[21] P. Dong *et al.*, "Reducing transport latency for short flows with multipath TCP," *J. Netw. Comput. Appl.*, vol. 108, pp. 20–36, Apr. 2018.
[22] S. Barré, "Implementation and assessment of modern host-based multipath solutions," Louvain School Eng., Univ. Catholique Louvain, Louvain-la-Neuve, Belgium, Tech. Rep., 2011. [Online]. Available: https://inl.info.ucl.ac.be/system/files/phd-thesis.pdf
[23] B. H. Oh and J. Lee, *Constraint-Based Proactive Scheduling for MPTCP in Wireless Networks*. Amsterdam, The Netherlands: North Holland, 2015.
[24] B. Y. L. Kimura, D. C. S. F. Lima, and A. A. F. Loureiro, "Alternative scheduling decisions for multipath TCP," *IEEE Commun. Lett.*, vol. 21, no. 11, pp. 2412–2415, Nov. 2017.
[25] D. Ni, K. Xue, P. Hong, and S. Shen, "Fine-grained forward prediction based dynamic packet scheduling mechanism for multipath TCP in lossy networks," in *Proc. Int. Conf. Comput. Commun. Netw.*, 2014, pp. 1–7.
[26] K. Xue, K. Chen, D. Ni, H. Zhang, and P. Hong, "Survey of MPTCP-based multipath transmission optimization," *J. Comput. Res. Develop.*, vol. 53, no. 11, pp. 2512–2529, 2016.
[27] N. Kuhn, E. Lochin, A. Mifdaoui, G. Sarwar, O. Mehani, and R. Boreli, "DAPS: Intelligent delay-aware packet scheduling for multipath transport," in *Proc. IEEE Int. Conf. Commun.*, Jun. 2014, pp. 1222–1227.
[28] F. Yang, Q. Wang, and P. D. Amer, "Out-of-order transmission for in-order arrival scheduling for multipath TCP," in *Proc. Int. Conf. Adv. Inf. Netw. Appl. Workshops*, 2014, pp. 749–752.
[29] S. Ferlin, Ö. Alay, O. Mehani, and R. Boreli, "BLEST: Blocking estimation-based MPTCP scheduler for heterogeneous networks," in *Proc. IFIP Netw. Conf.*, 2016, pp. 431–439.
[30] C. Paasch *et al.*, "Improving multipath TCP," Ph.D. dissertation, Louvain School Eng., Univ. Catholique de Louvain, London, U.K., 2014.
[31] S. Ro and D. N. Van, "Performance evaluation of MPTCP over a shared bottleneck link," *Int. J. Comput. Commun. Eng.*, vol. 5, no. 3, p. 176, 2016.
[32] C. Ling, W. Tang, P. Dong, W. Yang, X. Lou, and H. Zhou, "Blocking time-based mptcp scheduler for heterogeneous networks," in *Proc. Int. Conf. Cloud Comput. Secur.* Cham, Switzerland: Springer, 2018, pp. 364–375.
[33] Y.-S. Lim, E. M. Nahum, D. Towsley, and R. J. Gibbens, "ECF: An MPTCP path scheduler to manage heterogeneous paths," in *ACM SIGMETRICS*, 2017, pp. 147–159.
[34] S. Ferlin, Ö. Alay, D. A. Hayes, M. Welzl, and T. Dreibholz, "Revisiting congestion control for multipath TCP with shared bottleneck detection," in *Proc. IEEE INFOCOM Int. Conf. Comput. Commun.*, Apr. 2016, pp. 1–9.
[35] R. Barik, O. Alay, S. Ferlin, and M. Welzl, "LISA: A linked slow-start algorithm for MPTCP," in *Proc. IEEE Int. Conf. Commun.*, May 2016, pp. 1–7.

PINGPING DONG received the B.S., M.S., and Ph.D. degrees from the School of Information Science and Engineering, Central South University, China. She is currently a Teacher with the College of Information Science and Engineering, Hunan Normal University, Changsha, China. Her research interests include protocol optimization and protocol design in wide area networks and wireless local area networks.

JINGYUN XIE is currently pursuing the master's degree with the College of Information Science and Engineering, Hunan Normal University, Changsha, China. Her current research interest includes protocol optimization for heterogeneous networks.

WENSHENG TANG received the B.S. degree from Hunan Normal University, Changsha, China, in 1992, and the M.S. and Ph.D. degrees from the National University of Defense Technology, Changsha, in 1997 and 2009, respectively. He is currently a Professor with Hunan Normal University. His research interests include the protocol optimization and cloud computing.

NAIXUE XIONG received the Ph.D. degrees in sensor system engineering and dependable sensor networks from Wuhan University and the Japan Advanced Institute of Science and Technology, respectively. He is currently a Professor with the College of Intelligence and Computing, Tianjin University, China. Before he attended Tianjin University, he was with Northeastern State University, Georgia State University, the Wentworth Institute of Technology, and Colorado Technical University (a Full Professor for about 5 years) for about 10 years. His research interests include cloud computing, security and dependability, parallel and distributed computing, networks, and optimization theory.

HUA ZHONG is currently pursuing the master's degree with the College of Information Science and Engineering, Hunan Normal University, Changsha, China. His current research interest includes protocol optimization for heterogeneous networks.

ATHANASIOS V. VASILAKOS is currently a Professor with the Lulea University of Technology, Sweden. He served or is serving as an Editor for many technical journals, such as the IEEE TRANSACTIONS ON NETWORK AND SERVICE MANAGEMENT, the IEEE TRANSACTIONS ON CLOUD COMPUTING, the IEEE TRANSACTIONS ON INFORMATION FORENSICS AND SECURITY, the IEEE TRANSACTIONS ON CYBERNETICS, the IEEE TRANSACTIONS ON NANOBIOSCIENCE, the IEEE TRANSACTIONS ON INFORMATION TECHNOLOGY IN BIOMEDICINE, *ACM Transactions on Autonomous and Adaptive Systems*, and the IEEE JOURNAL ON SELECTED AREAS IN COMMUNICATIONS. He is also the General Chair of the European Alliances for Innovation.