 CMC, vol.57, no.2, pp.205-222, 2018

A MPTCP Scheduler for Web Transfer

Wenjun Yang[1], Pingping Dong[1, 2,*], Wensheng Tang[1], Xiaoping Lou[1], Hangjun Zhou[3], Kai Gao[4] and Haodong Wang[5]

Abstract: Multipath TCP (MPTCP) is the most significant extension of TCP that enables transmission via multiple paths concurrently to improve the resource usage and throughput of long flows. However, due to the concurrent multiple transfer (CMT) in short flow trans-mission, the congestion window (cwnd) of each MPTCP subflow is too small and it may lead to timeout when a single lost packet cannot be recovered through fast retransmission. As a result, MPTCP has even worse performance for short flows compared to regular TCP. In this paper, we take the first step to analyze the main reason why MPTCP has the diminished performance for short flows, and then we propose MPTCP-SF, which dynamically adjusts the number of subflows for each flow. In particular, MP TCP-SF firstly analyzes the distribution characteristics of the web objects to extract two thresholds to be used for classifying each flow. After eceiving each new ACK, MPTCP-SF periodically counts the data being sent based on per-flow and uses the threshold to classify the weblows. Finally, MPTCP-SF dynamically switches path scheduling model for different classification flows. We conduct extensive experiments in NS3 to evaluate its efficiency. Our evaluation proves that MPTCP-SF decreases the completion time of short flows by over 42.64% com-pared to MPTCP, and the throughput achieved by MPTCP-SF in transmitting long flows is about 11.11% higher than that of MPTCP in a WLAN/LTE wireless network. The results successfully validate the improved performance of MPTCP-SF.

Keywords: MPTCP, short flow, web transfer, timeout, path heterogeneity.

1 Introduction

Multipath transfer protocols enable data transmitting concurrently via multiple IP addresses or Network cards to provide high quality network service. It attracts a lot of interest and many researchers have begun to study the field of multipath transport protocols [Raiciu,

[1] College of Information Science and Engineering, Hunan Normal University, Changsha, 410081, Hunan, China.

[2] University of Science and Technology of China, Hefei, 230026, Anhui, China.

[3] Hunan University of Finance and Economy, Changsha, 410205, Hunan, China.

[4] Changsha University of Science & Technology, Changsha, 410114, Hunan, China.

[5] Department of Electrical Engineering and Computer Science, Cleveland State University, Cleveland, OH 44115, USA.

[*] Corresponding Author: Pingping Dong. Email: ppdong@csu.edu.cn.

Barre, Pluntke et al. (2011); Dong, Wang, Huang et al. (2016); Kheirkhah, Wakeman and Parisis (2016)]. IETF published MPTCP as an experimental standard in 2013. Compared to single-path TCP (SPTCP), MPTCP can greatly improve the network resource utilization and achieve higher throughput for long TCP flows [Raiciu, Barre, Pluntke et al. (2011)], the reason is that it distributes packets to all established subflows and it actively manages network congestion for all its subflows and shifts traffic from more to less congested paths.

It is well known that MPTCP has good performance for long-lived flows. For the short flows, around hundreds of KBs, MPTCP behaves worse compared against SPTCP [Deng, Netravali, Sivaraman et al. (2014); Kheirkhah, Wakeman and Parisis (2016)], because the packets are split to multiple subflows, the congestion window of a subflow may be very small over its life-time. Therefore, timeout will occur when a single lost packet cannot be recovered through the fast retransmission. However, in the real-time and interactive applications, such as web transfers, video streaming and online gaming, a majority of flows are short flows and often come with strict deadline regarding their completion time [Kheirkhah, Wakeman and Parisis (2016)]. If these short flows are not completed before their deadlines, some data will be lost and cause retransmission. Therefore, the overall network performance and the user's experience will be degraded harmfully. Then a major question raised is how to enhance MPTCP's performance in dealing with short flows while retaining its advantages on long flows.

There has been significant interest in designing an efficient algorithm for MPTCP. The methods proposed previously in literature can be categorized into two main aspects: Congestion control and path scheduling.

The first aspect mainly concentrates on shifting traffic from more congested paths to less congested ones. Currently some congestion control algorithms have been proposed: LIA (Linked Increases Algorithm) [Wischik, Raiciu, Greenhalgh et al. (2011a)], OLIA (Opportunistic Linked-Increases Algorithm) [Khalili, Gast, Popovic et al. (2012)], Balia (Balanced Linked Adaptation) [Peng, Walid, Hwang et al. (2016)], wVegas (Weighted Vegas) [Cao, Xu and Fu (2012)]. The performance of MPTCP is certain to be enhanced with these proposals, but these algorithms still unable to improve the efficiency of MPTCP when the flow size is small [Kheirkhah, Wakeman and Parisis (2016)].

The second aspect focuses on designing an efficient path scheduling algorithm for MPTCP to optimize the performance when dealing with short flows. Hwang et al. [Hwang and Yoo (2015)] designed a method to ensure short flows be quickly transmitted by fast paths if the network exists two paths and there are large difference of path delay between the slow and fast paths. BLEST [Ferlin, Alay, Mehani et al. (2016)] proposed a proactive scheduler which determines whether to send packets over the slow subflow or not. The first two scheduling policies rationally distribute packet into fast or slow path and efficiently reduce the completion time of short flow compared to the regular MPTCP. However, these two methods only perform well in the network situation where there are two paths. As proposed by MMPTCP [Kheirkhah, Wakeman and Parisis (2016)], combining Packet Scatter (PS) [Dixit, Prakash, Hu et al. (2013)] with MPTCP to transmit short flows and long flows

separately can effectively reduce the average completion time of short flows and increase the average throughput of long flows, but PS only performs well in the highly symmetrical topology [Raiciu, Barre, Pluntke et al. (2011)].

Based on the above analysis, we present MPTCP-SF (Multipath TCP for short flow) that is applicable to the scene where there are any number of subflows. The key idea behind MPTCP-SF is that MPTCP-SF periodically measures the amount of data being sent per-flow to identify the flow's classification, and then dynamically switches path scheduling model. The main contributions of this paper are as follows:

- We analyze the main factors that affect the performance of MPTCP in handling short flows.
- We propose a path scheduling model to eliminate these factors to achieve both performance improvement and latency reduction. Specifically, in order to be suitable for all web applications, we design a method which counts the data have been sent after each new ACK to identify the application's classification. And then we dynamically adjust path scheduling for different classification flows.
- We validate the efficiency of MPTCP-SF by conducting extensive experiments in different network scenarios. The simulations are based on our MPTCP-SF implementation in network simulator-3 (ns3). Compared with MPTCP, the test results demonstrate that MPTCP-SF greatly reduces the completion time of short flows in all scenarios.

The remainder of this paper is organized as follows. Section 2 discusses the related works. Section 3 introduces the design motivation of this paper. Section 4 describes the details of MPTCP-SF. Section 5 presents the evaluation of MPTCP-SF in ns3 with different experimental scenes. Finally, Section 6 concludes this paper.

2 MPTCP and related work

Multipath TCP is one of the significant multipath transport protocols. Its performance is certain to be affected by many factors [Paasch, Ferlin, Alay et al. (2014)]. Congestion control and path scheduling are two main aspects and have drawn considerable research attention.

Wischik et al. [Wischik, Raiciu, Greenhalgh et al. (2011); Khalili, Gast, Popovic et al. (2012); Peng, Walid, Hwang et al. (2016); Cao, Xu and Fu (2012)] proposed some algorithms for congestion control. LIA [Wischik, Raiciu, Greenhalgh et al. (2011)], O-LIA [Khalili, Gast, Popovic et al. (2012)], Balia [Peng, Walid, Hwang et al. (2016)], wVegas [Cao, Xu and Fu (2012)] are mainly dedicated to design an arithmetic model to control the increase of subflow's cwnd. Its design objective is balancing the degree of congestion among each subflow in network and improving the goodput when the network is bandwidth-hungry. The test results of these proposals prove that they all reduce the degree

of congestion and greatly improve the mean goodput compare to conventional MPTCP. However, these designs still do not take into account the existed problem when short flows are transported through multiple paths.

Next, the path scheduling policy is designed to rationally splitting data packets over multiple paths for optimizing the efficiency of default MPTCP scheduler. A freeze packet scheduling algorithm [Hwang and Yoo (2015)] was proposed to return the best subflow with the lowest RTT (Round Trip Time) for transfer data and freeze the slow path if the difference between the fast and slow paths is significant. BLEST [Ferlin, Alay and Mehani (2016)] proposed a proactive scheduler which estimates whether a path will cause HoL (head-of-line) blocking and determines whether to send packets over the slow subflow or not. However, both the algorithms in Hwang et al. [Hwang and Yoo (2015)] and Ferlin et al. [Ferlin, Alay, Mehani et al. (2016)] are designed for heterogeneous paths and only performed well in the network situation that there are two paths.

Recently, Kheirkhah et al. [Kheirkhah, Wakeman and Parisis (2016)] proposed a path scheduling algorithm named MMPTCP. MMPTCP achieves its objective by transmitting data in two phases. Initially, it follows the idea of PS [Dixit, Prakash, Hu et al. (2013)] to randomly scatter packets exploiting all available paths under a single congestion window. It is beneficial to short flow which has the strict deadline regarding their completion time. If the data being sent achieves a specific amount, the process of transport switches to regular MPTCP to deal with long-lived flows. However, PS is only suitable for the topology that traffic load is equal among servers in data center [Raiciu, Barre, Pluntke et al. (2011)], such as FatTree [Al-Fares, Loukissas and Vahdat (2008)] and VL2 [Greenberg, Hamilton, Jain et al. (2011)]. As long as the symmetry of topology is disrupted, PS will have a diminished performance for TCP flows.

3 Motivation

In this section, we study the main reason why MPTCP is not efficient for short flows, and then we present our design goals.

MPTCP manages a set of TCP connection, namely MPTCP subflow to transmit data simultaneously. It is obviously beneficial for long lasting flows. However, compared with SPTCP, if the data of short flow is divided into multiple paths, less traffic is allocated to each subflow. Therefore, the cwnd of each subflow will be maintained at a small value. This arrangement potentially impairs the transmission performance of short flows. When cwnd is less than three and a single packet is lost, the packet transmission cannot recover through fast retransmission because the sender cannot receive three duplicate ACKs and the timeout will occur. As a result, it eventually decreases cwnd of subflow and stalls the entire transmission flow.

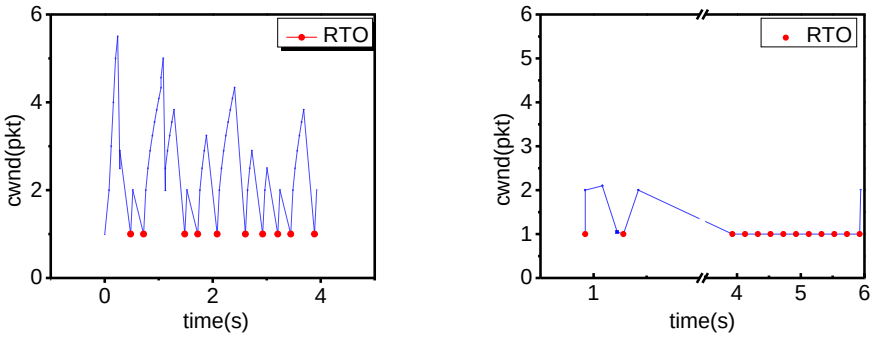
To validate the above discussed problems, we conduct a short flow transmission experiment to compare the performance of TCP with MPTCP that contains 8 subflows. The link characteristics for TCP and each MPTCP subflow are set as follows.

- TCP: Bandwidth=100 Mbit/s, Delay=10 ms and Loss=0.02%

- MPTCP: Bandwidth=100 Mbit/s, Delay=10−100 ms and Loss=0.02%

Fig. 1 shows the changing of congestion window over time t and the timeouts of short flow for TCP and MPTCP. Fig. 1(a) illustrates how the cwnd of TCP fluctuates and when timeout occurs in short flows, Fig. 1(b) shows how one of MPTCP subflows acts in the same short flow transmission.

As shown in Fig. 1, TCP has a faster cwnd growth rate than MPTCP subflow. The red dot in the graph indicates a timeout at that time. Clearly, we observed that the number of timeouts in Fig. 1(b) is 13, compare to 10 in Fig. 1(a). Whenever the timeout occurs, the value of cwnd decreases to 1, and it significantly diminishes the performance of MPTCP. In addition, Fig. 1(b) shows that, cwnd of the MPTCP subflow stays at 1 during the time from 4th s to 6th s, suggesting that there has no new packet allocated into this subflow. Thus the cwnd is unable to rise, which causes a number of retransmission timeouts (RTO) because the sender lacks the 3 duplicate ACKs. As a result, MPTCP with eight subflow has longer completion time for short flow than TCP.



(a) TCP (b) MPTCP with 8 subflows

Figure 1: Evaluation of cwnd changes between TCP and MPTCP which contains 8 subflows

Moreover, as researched in literature Oh et al. [Oh and Lee (2015); Yedugundla, Ferlin, Dreibholz et al. (2016); Dong, Yang, Tang et al. (2018)], MPTCP fails to provide satisfactory performance because of path heterogeneity that may cause packet reordering. In other words, If the gap between one path's RTT and the others is too wide, MPTCP has even worse performance than TCP. Therefore, reducing the heterogeneity between paths is also a problem we need to consider in our design.

Based on the above analysis, we conclude that timeout susceptibility and path heterogeneity are the main reasons for the poor performance of MPTCP compared to regular TCP in short flow transmission. The observation motivates us to design an efficient policy that dynamically schedules appropriate paths for each application to improve the performance of MPTCP in different scenarios.

Table 1: Size distribution characteristic of different web objects (KB)

Technology	end 2016	end 2015	end 2014
HTML	59	66	59
CSS	75	76	57
JavaScript	399	363	295
Images	1456	1443	1243
Flash	63	53	76
Total	2419	2262	1953

4 Protocol design

Based on the above discussion, a natural solution to the inefficiency of MPTCP when dealing with short flows is to dynamically adjust the number of MPTCP subflows for different flows. For example, in the case of short flows, it is better to utilize a single subflow. Unfortunately, a majority of applications do not render their flow size to the end-hosts [Kheirkhah, Wakeman and Parisis (2016)], i.e. the transport layer of MPTCP cannot get the high-level information and we cannot adjust the number of subflows according to the flow size.

Thus, in the paper, we take the first step to design a method that differentiates each application's classification. Firstly, we take a packet capture tool, named HttpWatch, to analyze the distribution law of web objects in internet. Then, according to the distribution characteris-tics of these web flows, we extract the threshold used for classifying each web applications. Finally, we propose an algorithm to adjust the number of subflows for different flows based on this classification.

4.1 MPTCP-SF and threshold design

To select the thresholds reasonably, we analyze the distribution characteristics of the web flows. The HTTP Archive conducts statistics on the distribution of different web objects (e.g. JavaScript, Image, HTML, Flash) based on 500,000 websites. The results are shown in Tab. 1. According to Tab. 1, the average size of the JavaScript is between 100 KB and 400 KB, the size of image is more than 400 KB, and the other types of objects are less than 100 KB. To further research the characteristics of network traffic, we analyze the distribution of web objects by HttpWatch based on six classic websites. The analysis results are shown in Fig. 2 which demonstrates that 90% of the object files are less than 100 KB. The proportion is about 10% of the files whose size is between 100 KB and 400 KB, and only a small part of the objects are larger than 400 KB.

Based on the above analysis of the network traffic distribution, it is reasonable to recognize 100 KB and 400 KB as the thresholds. It is worth noting that these thresholds are set according to the network traffic c haracteristics. T he r esearchers c an c hange t he values based on their requirements in practical applications.

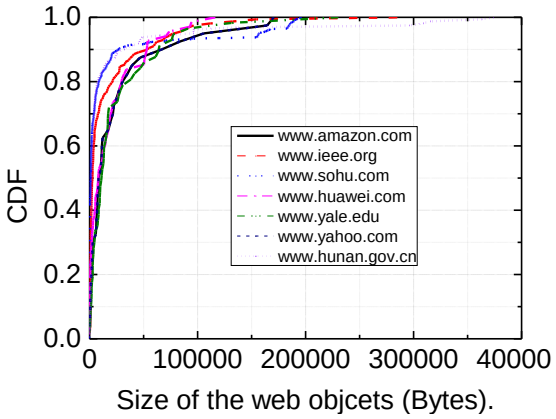


Figure 2: Distribution of web objects

4.2 Flow classification in real time

The flow size of some applications is uncertain, such as live video. Thus, to estimate the classification of flow F_i, we calculate the amount of data ($D_i(t)$) being sent after receiving each new ACK in real time (t). Specifically, if the value of $D_i(t)$ is less than 100 KB, F_i is classified as S_1. If value of $D_i(t)$ is between 100 KB and 400 KB, F_i is classified as S_2. Otherwise, F_i belongs to class S_3.

It is clear that we divide the flows into three classes, namely, S_1, S_2, S_3. In MPTCP-SF, flows of class S_1 are regarded as short flows. As analyzed above, choosing a single subflow to transport short flows is more efficient. Thus, flows of class S_1 are allocated to the fastest path. If the flows are classified as S_3, they can be considered as long-lasting flows and utilize all available subflows in MPTCP-SF design. For the flows classified in S_2, MPTCP-SF takes the path heterogeneity into account and uses RTT as the metric to select a subset of subflows, noted H_1 that contains a part of high quality utilized for flow transmission.

The variables used in this model are illustrated in Tab. 2.

4.3 MPTCP-SF and optimal subflow set H_1 design

If the RTT difference between subflows is too large, it will lead to the receiver buffer overflow and cause data loss. In this section, MPTCP-SF designs a parameter λ measuring the RTT difference between each subflow and merges the qualified subflows into subset H_1 which represents high quality paths will be utilized.

We first divide the n subflows into two groups: Theoptimal subflow group H_1 and the non-optimal subflow group H_2. There RTT differences are little among subflows in H_1 while RTTs of H_2 differ considerably. The details of the design of H_1 are illustrated as follows.

We use the variables N and N' to denote the set of all established subflows in the network and the sorted set of subflows in N by RTT. Initially, setting $N = \{p_1, p_2...p_r...p_n\}$ and $N' = \{p'_1, p'_2...p'_r...p'_n\}$ where p'_1 is the first subflow with minRTT (T_{min}) in N'. Let RTT_r

Table 2: Parameters and their physical significances in the MPTCP-SF algorithm

Parameter	Physical Significance
F_i	The *i*th flow
$D_i(t)$	The amount of data has been sent in time *t* based on F_i
n	The number of subflows
N	Set of all established subflows in network
p_r	The *r*th subflow in sets N
N'	New set of subflows after sorting subflows in set N by RTT
p'_1	The first subflow with minRTT in N'
p'_r	The rth subflow in sets N'
RTT_r	The RTT of *r*th subflow
H_1	The set which contains a part of high quality subflows
U_i	Set of selective subflows for F_i

Algorithm 1: H_1 design in MPTCP-SF

Input: A sorted set $N^{'} = p'_1, p'_2 ... p'_r ... p'_n$
Output: A set $H_1 = s_1, s_2 ... s_j (j \in [1, n])$ that selected for specific application

```
for each p'_r ∈ N' do
    if RTT_r/T_min ≤ λ then
        Put p'_r into set H_1;
    else
        Put p'_r into set H_2;
return H_1;
```

denote RTT of subflow r (p'_r) in N'. If RTT_r satisfies the following equation:

$$\frac{RTT_r}{T_{min}} \leq \lambda, \tag{1}$$

then we merge p'_r into the set of H_1. Otherwise, p'_r is allocated to H_2. The pseudo code of H_1 design is as shown in Algorithm 1. In addition, we conduct the corresponding experiments in Section 5.2 to validate the model and find the optimal value of λ.

4.4 The algorithm of MPTCP-SF

The basic idea of MPTCP-SF is dynamically adjusting the number of subflows for certain applications according to the flow classification in real time. The detailed description is illustrated as follows.

- If F_i belongs to S_1, we regard F_i as a short flow. Using multiple paths for short flow will cause the cwnd of each subflow to stay in a small value and increase slowly and

Algorithm 2: The proposed algorithm MPTCP-SF

Input: A set of subflows $N = \{p_1, p_2 ... p_r ... p_n\}$
Output: A set U_i that selected for F_i

Initialization at t=0
$D_i(t) = 0, U_i = \varnothing$
Schedule subflows into set U_i:
  Get the time t after receiving a new ACK;
  Order $p_r \in N$ by RTT and generate new Set N'= $\{p'_1, p'_2 ... p'_r ... p'_n\}$
  Calculate the value of $D_i(t)$ and Classify F_i
if F_i *belongs to* S_1 **then**
  $U_i = p'_1$;
else if F_i *belongs to* S_2 **then**
  $U_i = H_1$;
else
  $U_i = N'$;
return U_i;

greatly increase the completion time of short flows. Therefore, MPTCP-SF switches MPTCP to TCP by selecting only the fastest path for the flows in classes S_1, i.e. $U_i = p'_1$.

- If F_i belongs to S_2 and the RTT difference between multiple paths is large enough, using single subflow may decrease the average goodput while the usage of all subflows may cause the receiver buffer overflow. As a result, we design the set of *H1* which contains a part of qualified subflows to efficiently handle the applications in class S_2.

- If F_i belongs to S_3, MPTCP-SF immediately switches to default MPTCP scheduler to obtain higher throughput by for long flows.

Finally, we present the pseudo code of MPTCP-SF in Algorithm 2. It is called every time after a new ACK is received.

5 Performance evaluation

In this paper, the extensive experiments are conducted to validate the performance of MPTCP-SF based on ns3. Firstly, we test the basic performance of MPTCP, MPTCP-SF and TCP for short flows in different scenarios. Then, we conduct the experiments with different number of long-lasting flows to measure the average throughput of the three algorithms. Finally, we conduct the experiment for the scenario where both WLAN and LTE

interfaces are used to access internet, and are interacting with the MPTCP server. We investigate how short flows compete with long flows in this situation and select the average throughput and average flow completion time as the metrics.

5.1 Experiment setting

Following the network configurations in Raiciu et al. [Raiciu, Barre, Pluntke et al. (2011)], the number of subflows n ranges from 1 to 8. As shown in Fig. 3, the MPTCP clients and the server are connected by n routers, and therefore there are n MPTCP subflows in the internet. The size of short flow obeys the Pareto random distribution with the mean value of 57 KB according to the previous researches [Greenberg, Hamilton, Jain et al. (2011)]. The long flows are uniformly set to 15 MB for the statistical purpose. In the short flow and the long flow mixed scenario, the short flow (less than 100 KB, i.e. S_1 class) accounts for 90%. The flow that is larger than 100 KB and less than 400 KB (S_2 class) accounts for 10%. The number of long flow of 15 MB (S_3) is set to 1 according to the analysis in Fig. 2.

In the experiments, we set up two experimental scenarios, namely the high bandwidth with low latency (the Bandwidth is 100 Mbps and the delay ranges from 10 ms to 100 ms) and low bandwidth with high latency (the bandwidth 10 Mbps and the delay varies from 100 ms to 500 ms), respectively. The queue management mechanism of the routers adopts Droptail [Al-Fares, Radhakrishnan, Raghavan et al. (2010)], and the congestion control algorithm of the MPTCP protocol is RTT_Compensator [Wischik, Raiciu, Greenhalgh et al. (2011b)].

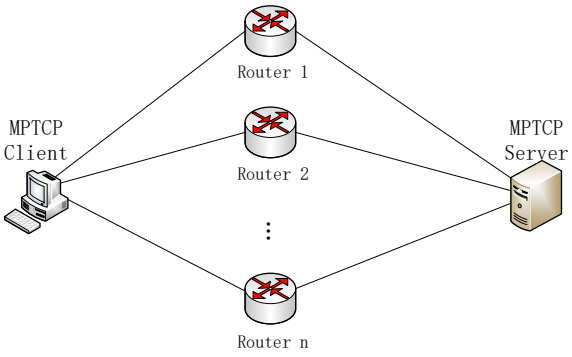


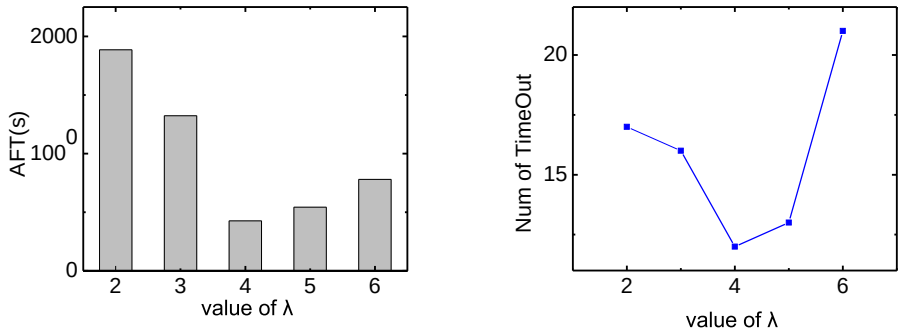
Figure 3: The experiment topology

5.2 Analysis for λ setting

We take the network scenario where n is set to 8 and the link packet loss rate is 0.02% as an example to analyze the effect of λ on MPTCP-SF. Fig. 4(a) shows the relationship between the value of λ and the Average Finish Time (AFT). Fig. 4(b) randomly selects a data flow with the size of 300 KB, and analyzes the relationship between the different value of λ and the number of timeout.

We can see from Fig. 4(a) that the minimum AFT can be obtained when λ is 4. Fig. 4(b) also demonstrates that the number of timeouts is the least at this time.

Based on the above analyses, we conclude that MPTCP-SF has the best performance when the value of λ is 4. Thus we set λ to 4 in the following experiments.



(a) The AFT with different λ (b) Timeout changes with the change of λ

Figure 4: Performance evaluation with different λ when there are 30 concurrent flows with size of 300 KB

5.3 Experimental results

In this section, we firstly study the average completion time of each algorithm with various network configurations to illustrate how each scheduler handles the challenges of short flows. And then, we measure the average throughput of long flows with the three algorithms. Finally, the performance of each algorithm in WLAN/LTE wireless network is investigated where long flows short flows are mixed.

5.3.1 The completion time of short flows

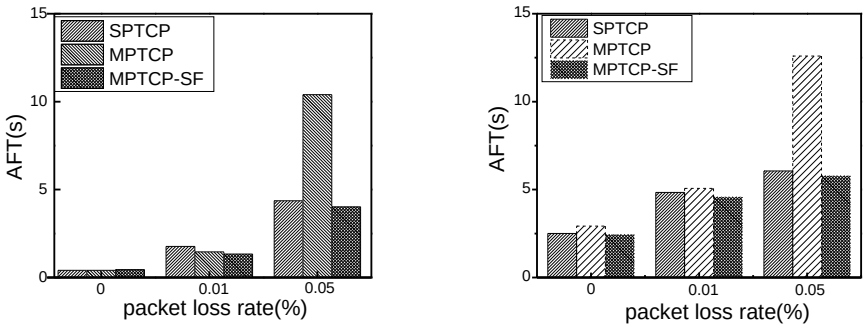
Lower completion time and higher transfer rate have always been goals pursued by researchers Lim et al. [Lim, Nahum, Towsley et al. (2017); Phejrsuksai and Pattaramalai (2017); Paasch, Detal, Duchene et al. (2012)]. In the network, short flows are required to meet the completion time before its strict deadline. Therefore, the AFT is one of the important indicators to measure the performance of short flow.

In order to validate the performance of each algorithm in short flow transmission extensively, we take both the different packet loss rates and the different concurrent numbers of flows into consideration.

Different packet loss rates. Fig. 5 is the AFT of 30 concurrent short flows under different packet loss rates. Among them, Fig. 5(a) represents the high bandwidth low delay network scenario where the bandwidth is 100 Mbps and the delay ranges from 10 ms to 100 ms

randomly. Fig. 5(b) represents the low bandwidth high delay network scenario where the bandwidth is 10 Mbps and the delay ranges from 100 ms to 500 ms randomly.

As can be seen from Fig. 5, AFT of the three algorithms increases with the increasing packet loss rate. In Fig. 5(a), when the packet loss rate is 0 and 0.01%, the corresponding AFT of the three algorithms is basically the same. While the packet loss rate of the router increases to 0.05%, MPTCP-SF greatly improves the performance in dealing with short flows compared with MPTCP. Its completion time is far less than MPTCP and close to TCP. In Fig. 5(b), the completion time of each algorithm increases compared to Fig. 5(a). Similarly, the performance of MPTCP drops most obviously, but MPTCP-SF performs almost as well as TCP.



(a) Bandwidth 100 Mbps (b) Bandwidth 10 Mbps

Figure 5: The average completion time of short flows with different packet loss rate

To further analyze the reason of the performance gains of MPTCP-SF in processing short-flows, we record how the cwnd changed for each algorithm in its life time, and count both the number of timeouts and the number of retransmissions. Fig. 6 shows a curve of the congestion window of subflow 1 when the packet loss rate is 0.05%. We can clearly see that the window of MPTCP and MPTCP-SF synchronously fluctuates in a zigzag manner between 0 s and 5 s. After 5 s, the congestion window of MPTCP drops to 1 and there is no window increasing process during this period. In comparison, the window of MPTCP-SF fluctuates frequently and the overall network throughput improves significantly. This is because MPTCP uses multiple subflows for short flows and is prone to RTO due to the small cwnd over each subflow. Once a timeout occurs, the sender's congestion window falls to a very small value and re-enters the slow start phase with MPTCP. For MPTCP-SF, a reasonable number of subflows is selected according to the flow classification and the cwnd is maintained at a large value. If packet loss occurs, it will quickly enter the fast retransmission and the cwnd can be recovered in a short time. That is the main reason why MPTCP-SF can efficiently reduce the completion time of short flows.

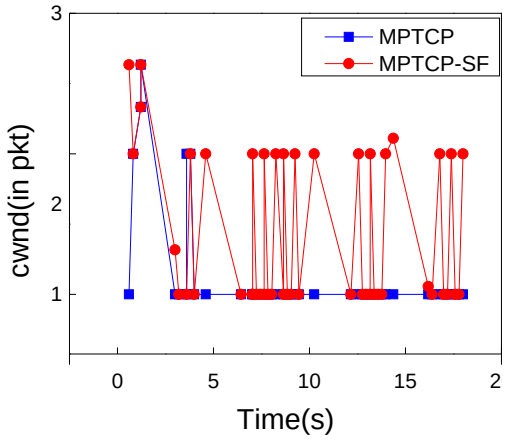


Figure 6: The cwnd changes over time

Table 3: The number of timeouts and retransmissions of MPTCP and MPTCP-SF

Transport protocol	Numbers of timeout	Numbers of retransmission
MPTCP	75	75
MPTCP-SF	56	56

The data in Tab. 3 indicates the number of timeouts and retransmissions of MPTCP and MPTCP-SF from 0th s to 20th s where the value of MPTCP is 75 and that of MPTCP-SF is 56. This further validates that that the MPTCP-SF can effectively reduce the occurrence of RTOs compared to MPTCP, and thus improves the transmission rate.

The number of different subflows. The number of subflows is one of the important factors affecting performance of multipath transport protocols. Fig. 7 shows the AFT of MPTCP and MPTCP-SF when using different number of subflows to transmit 30 concurrent short flows. The figure shows that the AFT of MPTCP rises sharply with the increasing number of subflows, while that of MPTCP-SF stay at a low value. This is benefited from the key idea of the MPTCP-SF that selecting different numbers of subflows for different flows. More specifically, MPTCP-SF can select the appropriate number of subflows to transmit the data flow regardless of the number of available subflows, causing the shorter the completion time and the higher the network utilization.

The mean throughput of long flows. For the long-lived flows, throughput is an important factor to measure the network performance. Fig. 8 compares the throughput of MPTCP, MPTCP-SF, and SPTCP when different numbers of long flows are transmitted concurrently. In this experiment, the high-bandwidth-low-delay scenario is adopted and the number of subflows n is 8. The average RTT of all subflows is 50 ms, and that of the SPTCP path is 50 ms.

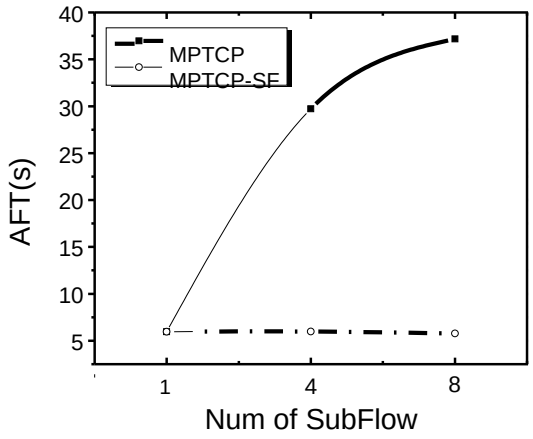


Figure 7: The effect of different number of subflows on short-flow AFT

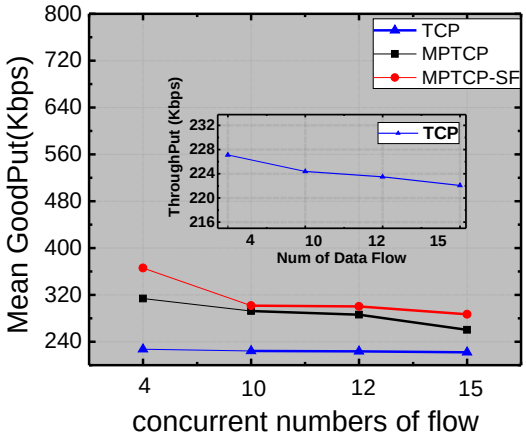


Figure 8: Throughput evaluation during long-flows transmission

As shown in Fig. 8, when the concurrent number of long-flow is less than 12, MPTCP-SF outperforms MPTCP as MPTCP-SF performs well in dealing with less-traffic. As more and more traffic floods, that is, the number of concurrent flows increases, the network becomes more congested and the obtained throughput decreases for each algorithm. However, both MPTCP and MPTCP-SF greatly improve the average throughput compared to regular TCP because of the benefit from transmitting data by multiple paths. Based on the above observation and analysis, the MPTCP-SF algorithm can dynamically change path scheduling policy and adapt well in long-flow transmissions. It greatly plays a high efficiency of

multipath transport protocol.

5.3.2 Performance analysis of long-short mixed flow

In the above experiments, we have compared and analyzed the performance of each algorithm for both short flows and long flows. These results prove that the MPTCP-SF takes the advantages of both TCP and MPTCP, and it can significantly improve the network performance during short flow and long flow transmissions.

In this subsection, we will further validate the performance of the MPTCP-SF algorithm when both short flows and long flow are present with the WLAN/LTE wireless network scenario as shown in Fig. 9.

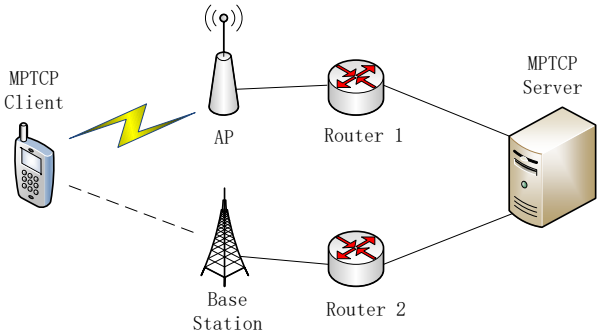


Figure 9: WLAN/LTE experiment scenario

In the WLAN/LTE network, the mobile device, namely the MPTCP Client accesses Internet to communicate with the MPTCP server via both WLAN and the LTE interfaces. MPTCP client can use the MPTCP protocol to establish two TCP connections to transmit data simultaneously. In this simulation, the default settings of parameters for WLAN and LTE are shown in Tab. 3. The starting time of application obeys Poisson distribution [Al-Fares, Radhakrishnan, Raghavan et al. (2010)].

Fig. 10 comprehensively illustrates the average completion time of the competing flows belonging to S_1, S_2, and S_3, respectively, where there are 30 concurrent long-short mixed flows. For short flows, TCP with the fastest connection is better than that of using multiple TCP connections [Deng, Netravali and Sivaraman (2014)]. And we use WIFI-TCP to denote the regular TCP with WIFI link for simplicity.

As shown in Fig. 10, the value of AFT corresponding to MPTCP-SF is almost the lowest. When the data flow is less than 100 KB (S_1 class), the MPTCP-SF reduces the AFT of the short flow by about 42.64% compared to the MPTCP which is closed to the AFT of WIFI-TCP. When dealing with the flows belonging to S_2 and S_3, the multipath protocol outperforms WIFI-TCP through the concurrent transmissions. In addition, the throughput gain achieved by MPTCP-SF in transmitting S_3 data is about 11.11% higher than that of MPTCP because MPTCP-SF is faster when transmitting head 400 KB of class S_3. This

further validate that MPTCP-SF optimizes the advantage of MPTCP with long-lived flows by CMT.

Table 4: Parameter setting of the link in wireless network

Simulation parameters	WLAN (IEEE 802.11a)	LTE
Channel bandwidth	20 MHz	60 MHz
Transmission power	25 dBm	eNB(40 dBm)/UE(20 dBm)
Propagation loss model	Fixed Rss Loss Model	Friis
Data rate/RB allocation	6 Mbps	DL(50)/UL(50)
Transmission delay	1 ms	10 ms

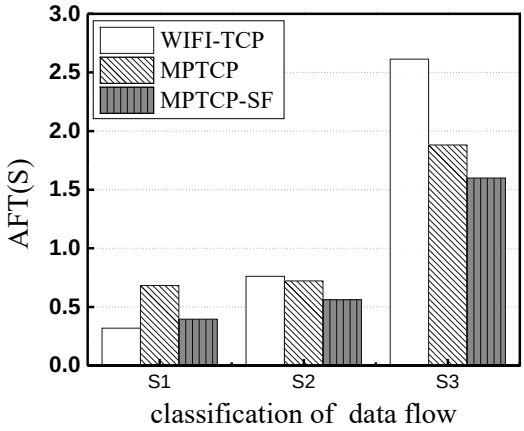


Figure 10: Performance evaluation in wireless network

From the above analysis, we can conclude that the MPTCP-SF reduces the overall average completion time and improves average throughput when dealing with long-short mixed flows in WLAN/LTE wireless network.

6 Conclusions and future work

Based on the characteristics of MPTCP, we analyze the problem of timeout susceptibility and path heterogeneity, which is harmful to the performance of MPTCP. To resolve these problems, we propose an MPTCP-SF algorithm that dynamically adjusts path scheduling for different flows and reduces the impact of path heterogeneity on MPTCP performance. And then, we conduct a series of experiments to compare the performance of SPTCP, MPTCP and MPTCP-SF for different flows under various scenarios. These experimental results demonstrate that MPTCP-SF can relieve the timeout occurrence of MPTCP, and retain the greater efficiency of MPTCP when dealing with long-lived flows. Besides, it can

also achieve higher performance when transmitting short flows, and thus greatly improves network resource utilization and the user experience.
In future work, we plan to implement MPTCP-SF in Linux kernel to improve the performance of MPTCP in real WAN and address some related practical issues.

Acknowledgemen: This work is supported by the National Natural Science Foundation of China (No. 61602171, 61602172), Scientific Research Fund of Hunan Provincial Education Department (No. 17C0960, 16C0050) and Hunan Provincial Natural Science Foundation of China (No. 2017JJ2016).

References

Al-Fares, M.; Loukissas, A.; Vahdat, A. (2008): A scalable, commodity data center network architecture. *ACM SIGCOMM Computer Communication Review*, vol. 38, no. 4, pp. 63-74.

Al-Fares, M.; Radhakrishnan, S.; Raghavan, B.; Huang, N.; Vahdat, A. (2010): Hedera: Dynamic flow scheduling for data center networks. *Nsdi*, vol. 10, pp. 19.

Cao, Y.; Xu, M.; Fu, X. (2012): Delay-based congestion control for multipath TCP. *20th IEEE International Conference on Network Protocols*, pp. 1-10.

Deng, S.; Netravali, R.; Sivaraman, A.; Balakrishnan, H. (2014): Wifi, ITE, or both? Measuring multi-homed wireless Internet performance. *Proceedings of the 2014 Conference on Internet Measurement Conference*, pp. 181-194.

Dixit, A.; Prakash, P.; Hu, Y. C.; Kompella, R. R. (2013): On the impact of packet spraying in data center networks. *Proceedings of IEEE*, pp. 2130-2138.

Dong, P.; Wang, J.; Huang, J.; Wang, H.; Min, G. (2016): Performance enhancement of multipath TCP for wireless communications with multiple radio interfaces. *IEEE Transactions on Communications*, vol. 64, no. 8, pp. 3456-3466.

Dong, P.; Yang, W.; Tang, W.; Huang, J.; Wang, H. et al. (2018): Reducing transport latency for short flows with multipath TCP. *Journal of Network and Computer Applications*, vol. 108, pp. 20-36.

Ferlin, S.; Alay, Ö.; Mehani, O.; Boreli, R. (2016): Blest: Blocking estimation-based MPTCP scheduler for heterogeneous networks. *IFIP Networking Conference and Workshops*, pp. 431-439.

Greenberg, A.; Hamilton, J. R.; Jain, N.; Kandula, S.; Kim, C. et al. (2011): Vl2: A scalable and flexible data center network. *Communications of the ACM*, vol. 54, no. 3, pp. 95-104.

Hwang, J.; Yoo, J. (2015): Packet scheduling for multipath TCP. *International Conference on Ubiquitous & Future Networks*, pp. 177-179.

Khalili, R.; Gast, N.; Popovic, M.; Upadhyay, U.; Le Boudec, J. Y. (2012): MPTCP is not pareto-optimal: Performance issues and a possible solution. *Proceedings of the 8th International Conference on Emerging Networking Experiments and Technologies*, pp. 1-12.

Kheirkhah, M.; Wakeman, I.; Parisis, G. (2016): Mmptcp: A multipath transport protocol for data centers. *35th Annual IEEE International Conference on Computer Communications*, pp. 1-9.

Lim, Y. S.; Nahum, E. M.; Towsley, D.; Gibbens, R. J. (2017): ECF: An MPTCP path scheduler to manage heterogeneous paths. *Proceedings of the 2017 ACM SIGMETRICS/International Conference on Measurement and Modeling of Computer Systems*, pp. 33-34.

Oh, B. H.; Lee, J. (2015): Constraint-based proactive scheduling for MPTCP in wireless networks. *Computer Networks*, vol. 91, pp. 548-563.

Paasch, C.; Ferlin, S.; Alay, O.; Bonaventure, O. (2014): Experimental evaluation of multipath TCP schedulers. *Proceedings of the 2014 ACM SIGCOMM workshop on Capacity sharing workshop*, pp. 27-32.

Peng, Q.; Walid, A.; Hwang, J.; Low, S. H. (2016): Multipath TCP: Analysis, design, and implementation. *IEEE/ACM Transactions on Networking*, vol. 24, no. 1, pp. 596-609.

Phejrsuksai, K.; Pattaramalai, S. (2017): Performance comparison of multipath TCP data transferring in bottleneck and disjoint-path wired networks connected with wi-fi. *Electrical Engineering Congress*, pp. 1-4.

Raiciu, C.; Barre, S.; Pluntke, C.; Greenhalgh, A.; Wischik, D. (2011): Improving datacenter performance and robustness with multipath TCP. *ACM SIGCOMM Computer Communication Review*, vol. 41, no. 4, pp. 266-277.

Wischik, D.; Raiciu, C.; Greenhalgh, A.; Handley, M. (2011): Design, implementation and evaluation of congestion control for multipath TCP. *NSDI*, vol. 11, pp. 8.

Yedugundla, K.; Ferlin, S.; Dreibholz, T.; Kuhn, N.; Hurtig, P. (2016): Is multipath transport suitable for latency sensitive traffic. *Computer Networks*, vol. 105, pp. 1-21.