Huang *et al. Journal of Cloud Computing: Advances, Systems and Applications* (2020) 9:13
https://doi.org/10.1186/s13677-020-00160-3

Journal of Cloud Computing:
Advances, Systems and Applications

RESEARCH Open Access

Tuning high flow concurrency for MPTCP in data center networks

Jiawei Huang[1], Weihe Li[1], Qi Li[1], Tao Zhang[1*], Pingping Dong[2] and Jianxin Wang[1]

Abstract

In the data center networks, multipath transmission control protocol(MPTCP) uses multiple subflows to balance traffic over parallel paths and achieve high throughput. Despite much recent progress in improving MPTCP performance in data center, how to adjust the number of subflows according to network status has remained elusive. In this paper, we reveal theoretically and empirically that controlling the number of concurrent subflows is very important in reducing flow completion time (FCT) under network dynamic. We further propose a novel design called MPTCP_OPN, which adaptively adjusts the number of concurrent subflows according to the real-time network state and flexibly shifts traffic from congested paths to mitigate the high tail latency. Experimental results show that MPTCP_OPN effectively reduces the timeout probability caused by full window loss and flow completion time by up to 50% compared with MPTCP protocol.

Keywords: Data center, MPTCP, Load balance, Congestion window

Introduction

With the rapid development of cloud computing and storage, data centers have gradually become the infrastructures of many bandwidth- and latency-sensitive applications such as web search and big-data analytics. Under the increasing traffic requirement of such applications, many multi-rooted tree topologies such as Clos [1] and Fat-tree [2] are deployed to utilize the multiple paths between the source and destination hosts to improve network throughput and reliability in data center.

In recent years, many state-of-the-art load balancing designs are proposed to make full use of the available multiple paths. As the de facto load balancing scheme in data center, Equal Cost Multipath (ECMP) [3] randomly sends each flow to one of the parallel paths by hash function. Though ECMP is very simple, it suffers from well-known performance problems such as hash collisions and low link utilization. Random Packet Spraying (RPS) [4] is a packet-level load balancing scheme that sprays each packet to a random path to obtain high link utilization, while also leading to packet reordering and throughput loss. Compared with the flow-level and packet-level load balancing schemes, MultiPath TCP (MPTCP) uses parallel subflows as the rerouting granularity to achieve better tradeoff in minimizing packet reordering and increasing link utilization [5].

Unfortunately, existing MPTCP designs do not consider an important issue, that is, the number of subflows has significant impact on MPTCP performance. On the one hand, if the number of subflows is small, the network resources of multiple paths are wasted, resulting in suboptimal flow completion time and even hot spots in congested paths. On the other hand, if the number of subflows is large, the congestion window of each subflow becomes very small, easily leading to timeout (i.e., full window loss) and high tail latency under heavy congestion [6–10].

In this work, we propose a novel design called MPTCP_OPN, which adaptively adjusts the number of concurrent subflows according to the real-time network state. Under heavy congestion, MPTCP_OPN shrinks the number of subflows to avoid timeout event caused by full window loss. On the contrary, MPTCP_OPN utilizes more subflows to improve the link utilization. Moreover, when some subflows experience severe congestion, MPTCP_OPN flexibly shifts the unlucky subflows from congested paths to less congested ones to mitigate the high tail latency.

*Correspondence: taozhang@csu.edu.cn
[1]School of Computer Science and Engineering, Central South University, 410083 Changsha, China
Full list of author information is available at the end of the article

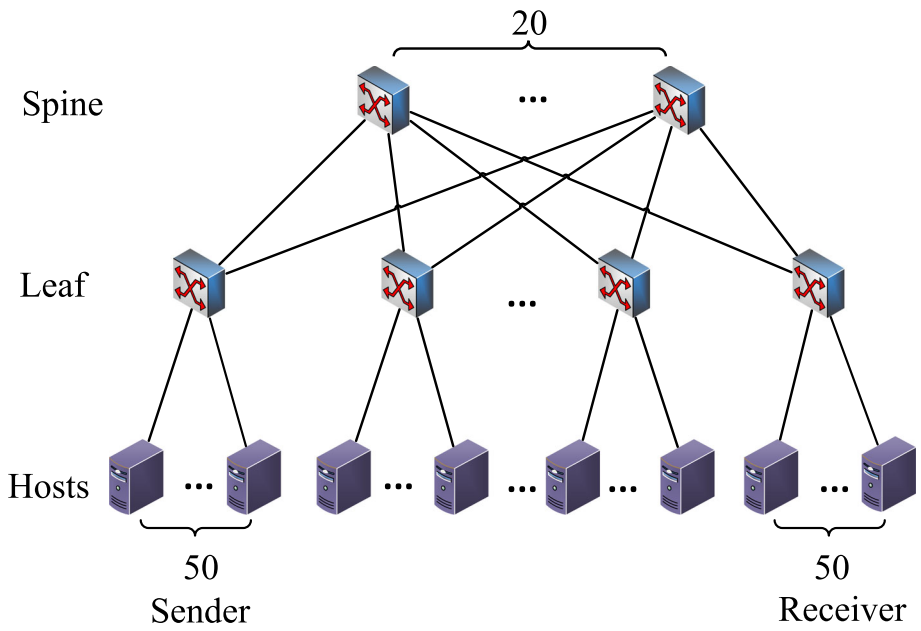

Fig. 1 Leaf-spine topology

We briefly summarize the contributions of this paper, as follows:

- We provide extensive study to exploit the impact of number of subflows on MPTCP performance. Through empirical studies and theoretical analysis, we reveal that controlling the number of concurrent subflows is crucial to achieve the tradeoff between high link utilization and low tail latency under network dynamic.
- We propose a novel design MPTCP_OPN by carefully adjusting the number of concurrent subflows on the sender-side. To reduce the long tail flow completion time due to the stalled subflow, MPTCP_OPN transfers traffic from congested paths to less congested ones. The design only needs to be deployed at the sender-side without any changes made to switch, thus ensuring the minor deployment overhead.
- We evaluate our design on large-scale test. The results demonstrate that, MPTCP_OPN is able to greatly reduce the flow completion time by up to 50% compared with the state-of-the-art load balancing designs, such as RPS, MPTCP and MMPTCP.

The remainder of this paper is structured as follows. In "Design motivation" section, we describe our design motivation. The design detail of MPTCP_OPN is presented in "MPTCP_OPN design" section. In "Performance evaluation" section, we show the experimental results of NS2 simulation. In "Related works" section, we demonstrate existing approaches. Finally, we conclude the paper in "Conclusion" section.

Design motivation

A MPTCP connection consists of multiple subflows, each of which may take a different route across available paths and maintains its own congestion window. In this section, we use NS2 simulation to evaluate the impact of the number of MPTCP subflows on the congestion window size, number of timeout events and flow completion time.

Firstly, we measure the congestion window of subflows under different number of subflows. The experimental scenario uses the Leaf-Spine topology as shown in Fig. 1. The number of available paths between any pair of hosts is 20. The link bandwidth and round trip time are 1Gbps and 100μs, respectively. Under each ToR switch, there are 50 hosts, each of which sends one MPTCP flow to one receiver under the other ToRs. The RTO of each subflow is set to 200ms as default. The default flow size of MPTCP is 800 packets with 1.5KB packet size. In order to obtain the average test results, the experiments are repeated for 20 times.

Figure 2 shows the average congestion window sizes of subflows with varying number of subflows from 1 to 20. The size of each MPTCP flow increases from 400 to 800 packets. When the number of subflows increases, the average congestion window of subflows is reduced. The reason is that, when the sender uses more subflows, the amount of data transmitted by each subflow decreases. For example, when 400 packets are transferred by 20 subflows, each subflow transmits only 20 packets on average, resulting in the average congestion window as small as about 5.

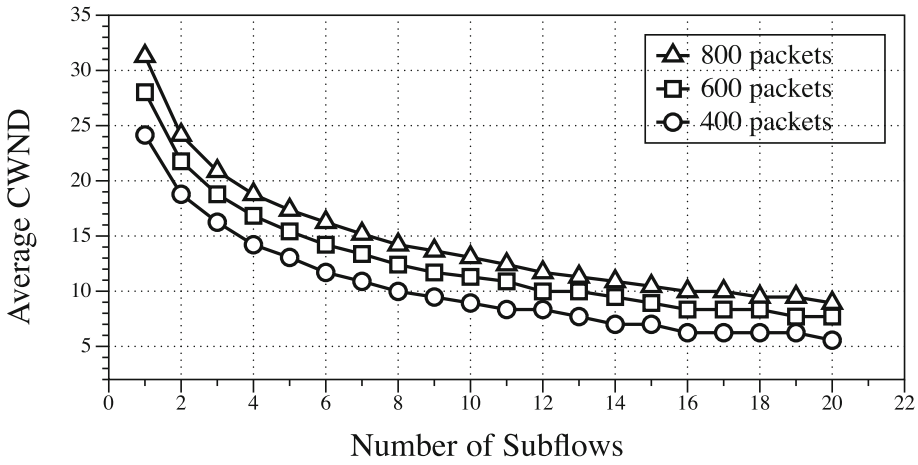

Fig. 2 Average congestion window with varying number of subflows

Secondly, we measure the number of timeout events with varying number of subflows. Under heavy congestion, if multiple packet losses occur in a single flow, this unlucky flow may experience full window loss and induce timeout. Figure 3 shows the maximum, minimum and average number of timeout events. Since the congestion window decreases with larger number of subflows, each subflow is more likely to lose packets within the entire congestion window, resulting in more timeout events. This will make the sender unable to rely on fast retransmission to recover lost packets, and has to wait for the retransmission timeout (RTO) (i.e, 200ms) to retransmit the lost packets.

Finally, we calculate the average flow completion time of MPTCP under different number of subflows. Here, the flow completion time of MPTCP flow is the time when the last subflow finishes its transfer. Figure 4 shows the average flow completion time for 20 experiments, respectively.

From Fig. 4, we observe that when the number of subflows is small, the flow completion time decreases with larger number of subflows, showing the benefit of high link utilization achieved by more subflows. However, when the number of subflows exceeds a certain point, the small congestion window of each subflow easily leads to higher timeout probability, which enlarges the tail latency and greatly increases the overall flow completion time.

Our observation of this experiment leads us to conclude that (i) large number of subflows decreases the congestion window, leading to the significantly increment in timeout probability under heavy congestion, and (ii) using small number of subflows reduces link utilization and enlarges the flow completion time under light congestion, which suggests that adopting fixed number of subflows is not an optimal solution under dynamic network traffic. These conclusions motivated us to investigate a novel approach

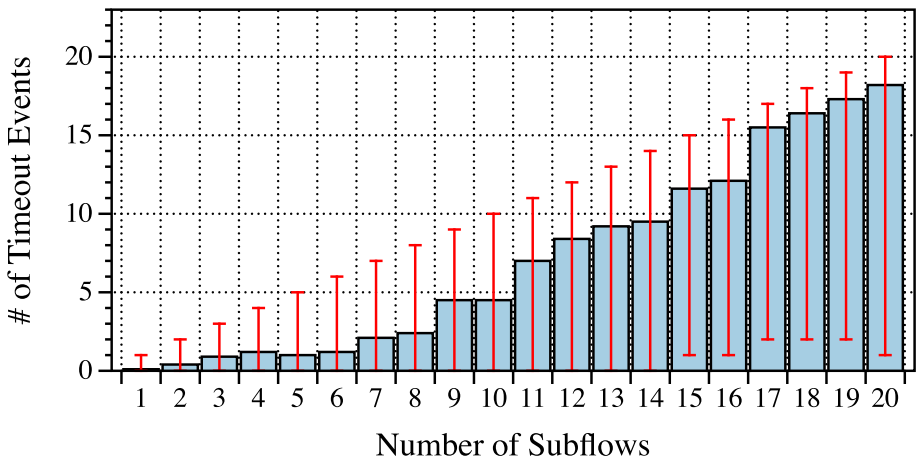

Fig. 3 Timeout events with varying number of subflows

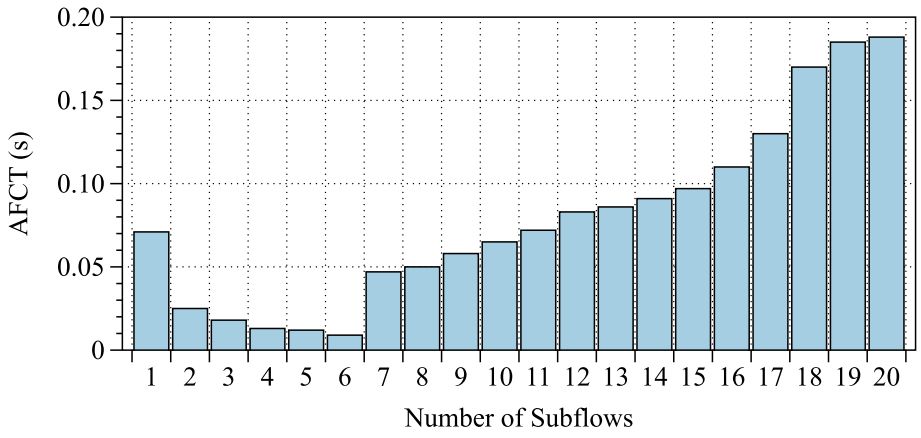

Fig. 4 Average flow completion time

adjusting number of subflows. In the rest of this paper, we present our MPTCP_OPN design in detail.

MPTCP_OPN design

The objective of MPTCP_OPN is to help MPTCP to achieve high performance in dynamic network. The core of MPTCP_OPN design is to adjust the number of subflows (i.e., flow concurrency) based on network state to achieve the tradeoff between link utilization and low tail latency. Besides, to mitigate the impact of network uncertainties such as link failures, MPTCP_OPN swiftly moves the stalled traffic from the slow path to fast ones once detecting congestion. Therefore, we design MPTCP_OPN consisting the Concurrency Tuning and Traffic Shifting modules, which only modify the MPTCP scheduler on the sender side.

Concurrency tuning

Tuning the concurrency of subflows is the core design of MPTCP_OPN. To address this issue, we establish the MPTCP throughput model to solve the optimal number of subflows.

We analyze the evolution of total congestion window of multiple subflows to obtain the aggregated throughput. Table 1 shows the variables used in the modeling analysis. Figure 5 depicts the total congestion window CW_n of n subflows. We use i and d to denote the window-increasing and window-decreasing factor of the congestion window, respectively. If no packet loss occurs, the total window size of n subflows will increase by ni in each round trip time (RTT). On the contrary, once packet loss happens, one of the subflows will reduce its congestion window, while the window size of the other $n-1$ subflows remaining unchanged. Therefore, the total congestion window becomes $(n-d)CW_n/n$. After packet loss occurs, it requires $dCW_n/(n^2 i)$ RTTs to restore the total window to CW_n.

Given the packet loss rate p, the total amount of packets sent by n subflows is $1/p$ in the period of window-increasing. Thus, we get

$$\frac{dCW_n}{n^2 i} * \frac{(n-d)CW_n}{n} + \frac{1}{2} * \frac{dCW_n}{n^2 i} * \frac{dCW_n}{n} = \frac{1}{p}. \quad (1)$$

Then, the total congestion window CW_n of n subflows is

$$CW_n = n\sqrt{\frac{2ni}{(2n-d)pd}}. \quad (2)$$

Therefore, the average congestion window size for each subflow is

$$W = \sqrt{\frac{2ni}{(2n-d)pd}}. \quad (3)$$

For each subflow, the timeout probability can be calculated as the probability that the subflow loses all the packets in its congestion window W, that is p^W. Then for a MPTCP flow with n subflows, the probability that at least one subflow has a full window loss is

$$P_{TO} = 1 - \left(1 - p^W\right)^n. \quad (4)$$

Table 1 Symbols

Symbol	Definition
n	The number of subflows
CW_n	Total congestion window size of n subflows
W	Average congestion window size of one subflow
d	Window-decreasing factor, default is 0.5
i	Window-increasing factor, default is 1
p	Packet loss rate
RTT	Round trip time
T	Average throughput of n subflows
P_{TO}	Timeout probability

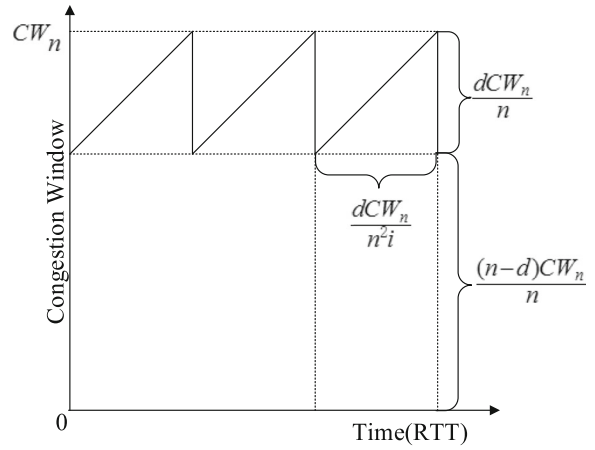

Fig. 5 Total congestion window of concurrent subflows

Substituting formula (3) into formula (4), we have

$$P_{TO} = 1 - (1 - p^{\sqrt{\frac{2ni}{(2n-d)pd}}})^n. \tag{5}$$

Taking into account the impact of timeout, the total throughput T of n subflows can be obtained as

$$T = \frac{(1/p)}{(dCW_n/n^2 i)} * \frac{MSS}{RTT} * (1 - P_{TO}). \tag{6}$$

Substituting formula (2) and (5) into formula (6), we have

$$T = \sqrt{\frac{ni(2n-d)}{2pd}} * \frac{MSS}{RTT} * \left(1 - p^{\sqrt{\frac{2ni}{(2n-d)pd}}}\right)^n. \tag{7}$$

Figure 6 shows the change of throughput as the number of subflows increases under different packet loss rate. The modeling result shows that, under the small number of subflows, more subflows are beneficial in improving throughput due to higher link utilization. On the contrary, when the number of subflows is large, the small congestion window of each subflow leads to high timeout probability and throughput degradation.

Therefore, it is clear that we can obtain the optimal number of subflows n^* to achieve the maximum throughput and the minimum flow completion time. To simplify the calculation, we use symbol a to denote

$$a = \frac{MSS}{\sqrt{2pd}RTT}. \tag{8}$$

We calculate the derivative of T as

$$\begin{aligned}\frac{dT}{dn} = a[&\log\left(1 - p^{\sqrt{\frac{4n}{p(2n-\frac{1}{2})}}}\right)\left(1 - p^{\sqrt{\frac{4n}{p(2n-\frac{1}{2})}}}\right)^n \\ &- \frac{np^{\sqrt{\frac{4n}{p(2n-\frac{1}{2})}}}\log(p)\left\{\frac{4}{p(2n-\frac{1}{2})} - \frac{8n}{[p(2n-\frac{1}{2})]^2}\right\}(1-p^{\sqrt{\frac{4n}{p(2n-\frac{1}{2})}}})^{n-1}}{2*\sqrt{\frac{4n}{p(2n-\frac{1}{2})}}}] * \sqrt{n\left(2n-\frac{1}{2}\right)} \\ &+ \frac{a\left(4n-\frac{1}{2}\right)\left(1-p^{\sqrt{\frac{4n}{p\left(2n-\frac{1}{2}\right)}}}\right)}{2*\sqrt{n\left(2n-\frac{1}{2}\right)}}.\end{aligned} \tag{9}$$

Since $\sqrt{2n-\frac{1}{2}}$ and $\sqrt{2n}$ are similar within the range of n, formula (9) can be reduced to

$$\frac{dT}{dn} \approx \sqrt{2}a\left[\log\left(1 - p^{\sqrt{\frac{2}{p}}}\right)\left(1 - p^{\sqrt{\frac{2}{p}}}\right)^n\right] * n + \frac{4an\left(1 - p^{\sqrt{\frac{2}{p}}}\right)}{2\sqrt{2} * n}. \tag{10}$$

When $\frac{dT}{dn}$ is equal to 0, we get the optimal number of subflows n^* as

$$n^* = abs\left[\frac{lambertw\left(0, p^{\sqrt{\frac{2}{p}}} - 1\right)}{\log\left(1 - p^{\sqrt{\frac{2}{p}}}\right)}\right]. \tag{11}$$

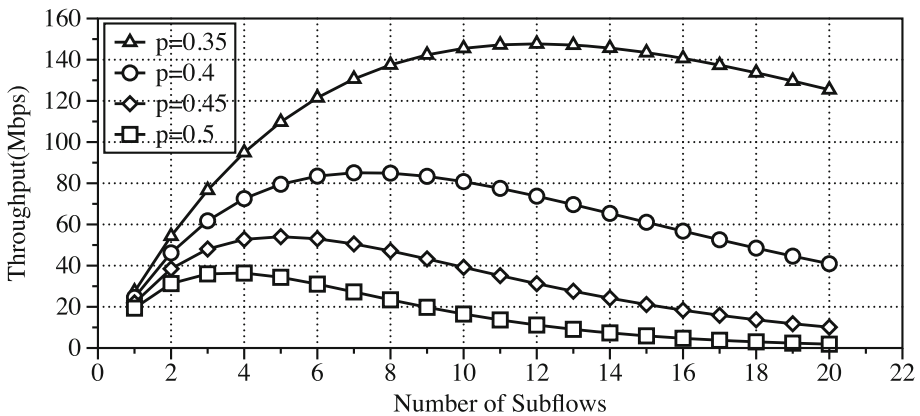

Fig. 6 Throughput with varying number of subflows

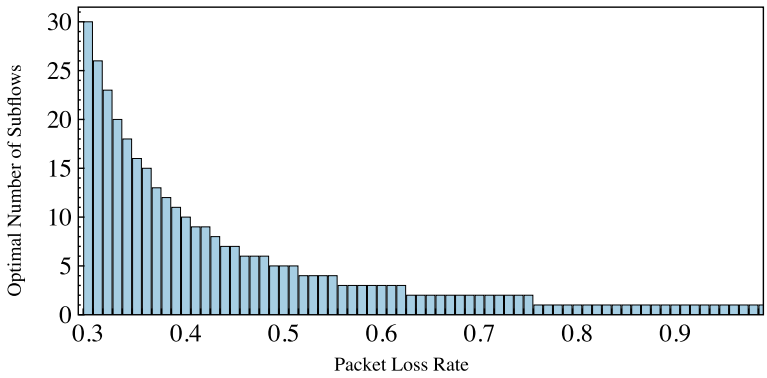

Fig. 7 Optimal number of subflows with varying packet loss rate

lambertw is a Lambert W function, which is the inverse relation of function $f(x) = xe^x$, where e^x is the exponential function, and x is any complex number. And *abs* is a function which uses to calculate the complex magnitude.

In order to reduce the computational complexity and increase the feasibility of protocol deployment, we first perform offline calculations by enumerating the packet loss rate and solving the corresponding result of n^*. Then, in the online phase, we can use a table to search n^* according to packet loss rate. Figure 7 shows the relationship between the packet loss rate and n^*. The granularity of packet loss rate is 0.01. Due to the constraint of the OS kernel [11], the maximum number of n^* is 32.

Traffic shifting

In production data center, there widely exists a series of uncertainties, such as link failures, traffic dynamic, and heterogeneous switching equipments. These uncertainties can make some links slow or unavailable. If some unlucky subflows are scheduled on these bad paths, the flow completion time is enlarged [12]. To mitigate the high tail latency problem caused by the dynamic network uncertainties, we propose a Traffic Shifting mechanism, which quickly detects the bad paths and moves the traffic to the good paths.

Specifically, MPTCP_OPN leverages the information of ACK packets to detect the link failures and traffic congestions. The sender maintains a timer to periodically detect the path status. If the sender has not received any ACK packets on a path when the timer expires, we consider that link failures and traffic congestions occur on that path. By this way, we classify all the available paths into two categories: FAST and SLOW. The SLOW path means that link failures and traffic congestions have occurred on it. Otherwise, it is identified as a FAST path. Moreover, to avoid the long tail latency, the sender employs the FAST path to retransmit the unacknowledged packets on the SLOW path.

Figure 8 depicts the design of the traffic shifting strategy. Figure 8a shows that, when the multiple paths are symmetric, the MPTCP sender quickly transmits data on three good paths. As shown in Fig. 8b, however, if path 1 becomes congested, although each subflow utilizes the window-based congestion control algorithm to achieve load balancing between paths, a few packets are still transmitted on path 1, causing the long tail delay of overall transmission. In our design, MPTCP_OPN scheme completely avoids the congested path. As shown in Fig. 8c, when the MPTCP_OPN sender detects the slow path, it will no longer use the slow path to send subflow. Moreover, the on-flight data packets whose corresponding ACK packets have not arrived at the sender within a period of time (i.e., 2RTTs) are retransmitted immediately to other good paths. Therefore, the receiver need not wait for the stalled on-flight packets, thus mitigating the long tail latency of subflow on congested path.

To quickly obtain the global link status and maintain high network utilization, we set the initial number of subflows as the number of spine switches. Then, the number of subflows varies according to the measured packet loss rate. To explicitly depict the procedure of subflow adjustment, we draw the diagram as shown in Fig. 9.

The initial number of subflows is set as the number of spine switches m to quickly obtain the global link status and maintain high network utilization. Then, the sender assigns different five-tuples to all the subflows for hashing them on available paths with ECMP. We use s and l to respectively record the number of packets have been sent and the number of dropped packets periodically. The duration time of the period is determined by a timer *timer*_3 to periodically calculate packet loss rate η. In our design, *timer*_3 is set to 10ms based on Ref. [13]. When *timer*_3 is expired, the packet loss rate η is calculated by $\eta = l/s$. Then, the optimal number of subflows n^* is obtained based on η through looking up the precomputed table.

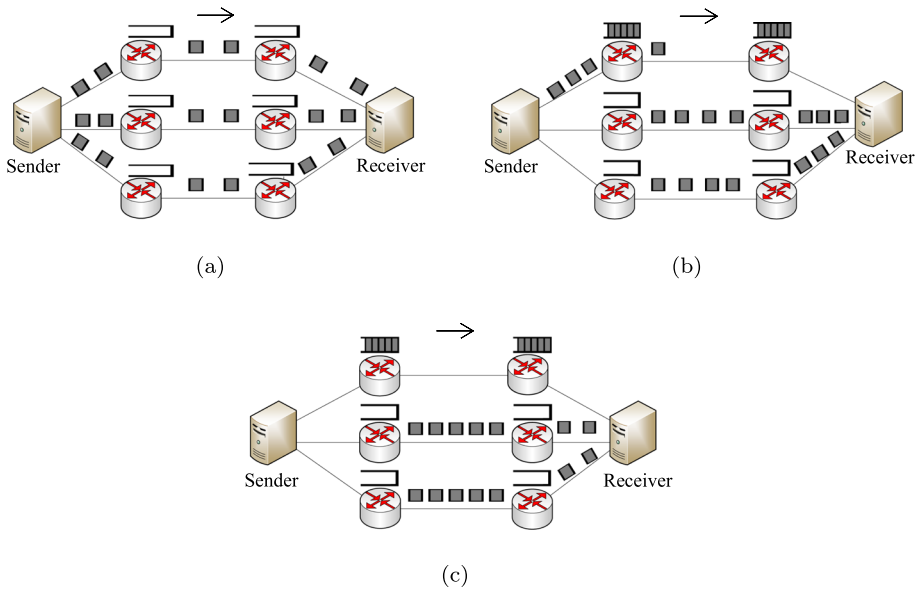

Fig. 8 Traffic Shifting Mechanism. **a** MPTCP under path symmetry; **b** MPTCP under path asymmetry; **c** MPTCP OPN under path asymmetry

The main operations of MPTCP_OPN are shown in Algorithm 1. First, the sender periodically detects the path status. As shown in lines 2-4, if the sender has not received an ACK packet on a path until the status-monitoring timer timer_1 expires, the path is marked as SLOW. Secondly, in order to avoid the long tail latency due to the blocked on-flight packets on slow path, when the sender detects a slow path, it will quickly retransmit the unacknowledged packets to other fast paths as shown in line 5. Finally, as shown in lines 15-20, the sender obtains the optimal number of subflows n^* based on the real-time network status. If n^* is smaller than the current number of fast paths, the flow from the upper layer is cut into n^* subflows. Otherwise, the flow uses all fast paths to transfer data. In the path detection, the timeout value δ of status-monitoring timer timer_1 is set as 2 times of the average round trip time. According to the previous studies, such as Ref. [14, 15], traffic in data center is highly dynamic. Congestion can promptly happen when a few high-rate flows start, and it can disappear as they finish. Therefore, we periodically reset the status of all the paths when timer 2 expires after ε. According to the default TCP timeout period in the current operating system, the timeout value ε of state-resetting timer timer_2 is set to 200ms.

Algorithm 1

Initialization: $\forall p_i$.mode←FAST; path[ ]←all paths; $\delta \leftarrow 200\mu$s; $\varepsilon \leftarrow$200ms;

1: **When the timer_1 expires after** δ:
2: **for** each path $p_i \in$path[ ] **do**
3: **if** none ACKs are received on p_i **then**
4: p_i.mode←SLOW;
5: retransmit the unacknowledged packets on fast paths;
6: **end if;**
7: **reset the timer_1;**
8:
9: **When the timer_2 expires after** ε**:**
10: **for** each path $p_i \in$path[ ] **do**
11: p_i.mode←FAST;
12: **reset the timer_2;**
13:
14: **On receiving a new flow from the upon layer:**
15: Obtain the optimal number of subflows n^*;
16: **if** $n^* <$ the number of fast paths **then**
17: transmit the flow on randomly selected n^* fast paths;
18: **else**
19: transmit the flow on all fast paths;
20: **end if;**

Performance evaluation

We evaluate the performances of MPTCP_OPN, ECMP, RPS, MPTCP, and MMPTCP through large-scale NS2 simulation tests.

- **MPTCP_OPN:** To balance traffic over parallel links and achieve high throughput, MPTCP_OPN adaptively adjusts the number of subflows according to network status.
- **ECMP:** As the de facto multipath load balancing mechanism, ECMP is broadly deployed on

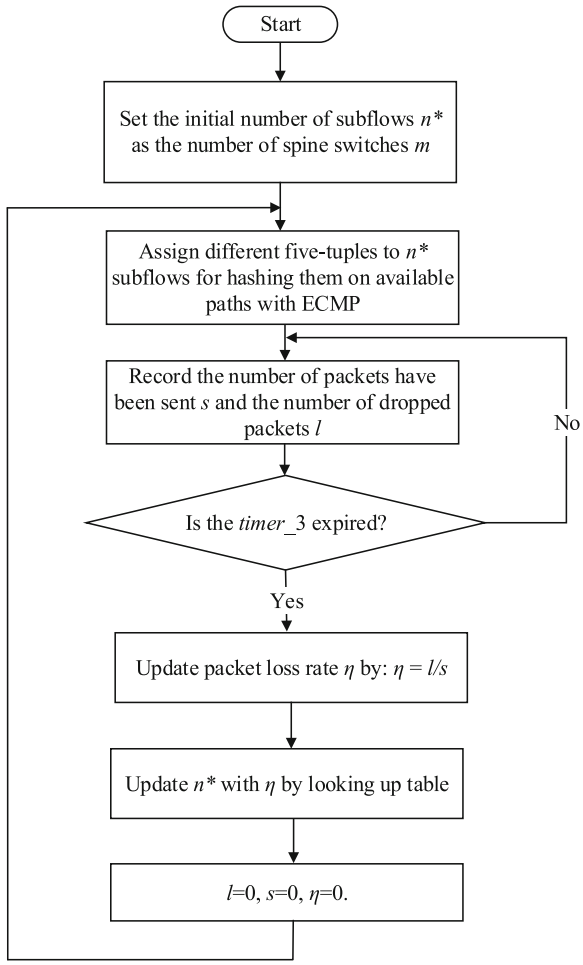

Fig. 9 Subflow adjustment

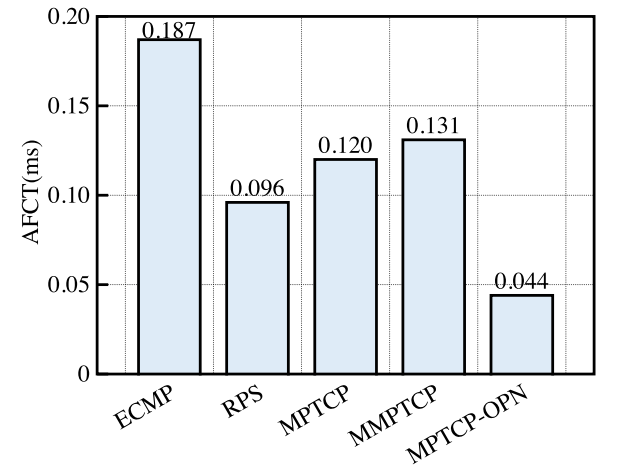

Fig. 10 AFCT of homogenous flows

commodity switches in data center networks. It randomly hashes each flow to one of the available paths based on the five-tuple of TCP packet header.

- **RPS:** RPS is a packet-level mechanism which randomly sprays packets to all available paths between any pairs of source and destination hosts.
- **MPTCP:** MPTCP uses fixed number of subflows to achieve better link utilization and alleviate packet reordering.
- **MMPTCP:** MMPTCP adopts different policies for long and short flows. MMPTCP utilizes stationary number of subflows to provide high throughput for long flows. Meanwhile, to reduce the flow completion time of short flows, RPS is employed to swiftly utilize available link resources.

As shown in Fig. 1, the tests use the Leaf-Spine topology with 40 ToR and 20 Core switches. There are 50 hosts under each ToR switch and 20 available equivalent paths between each pair of hosts. The switch buffer size is 256 packets. The link bandwidth and latency are respectively 1Gbps and 100μs, and RTO is set to 200ms. For ECMP and RPS, DCTCP is chosen as the transport protocol [16]. The experiments use flow completion time and throughput as performance metrics.

Basic performances

First, we test the basic performance of 1000 flows with same size between each pair of hosts. The flow size is set to 200 packets with 1.5 KB packet size. Figure 9 shows the test results of average flow completion time.

As shown in Fig. 10, ECMP experiences the largest flow completion time. The reason is that, ECMP uses the flow as the path-switching granularity, without making full use of the available multiple paths. Moreover, the hash collision easily leads to congestion hot spots, which increase the flow completion time. RPS chooses the next hop for each packet to leverage multiple paths, thus greatly improving link utilization. However, RPS easily causes packet reordering problem. Once receiving three or more duplicate ACKs due to disordered packets, the sender triggers unnecessary retransmission and mistakenly reduces its congestion window, resulting in suboptimal flow completion time. Both MPTCP and MMPTCP use all paths to send subflows to achieve high link utilization and mitigate packet reordering. However, when some subflows experience the heavy packet loss (i.e., full window loss) on congested paths, the overall transmission efficiency is reduced. Compared with the other schemes, based on the real-time network status, MPTCP_OPN adjusts the number of subflows to tackle the serious congestion, effectively reducing the flow completion time.

Next, we change the flow size to test the performance under heterogeneous traffic, which is mixture of long and short flows subjected to heavy-tailed distribution. The

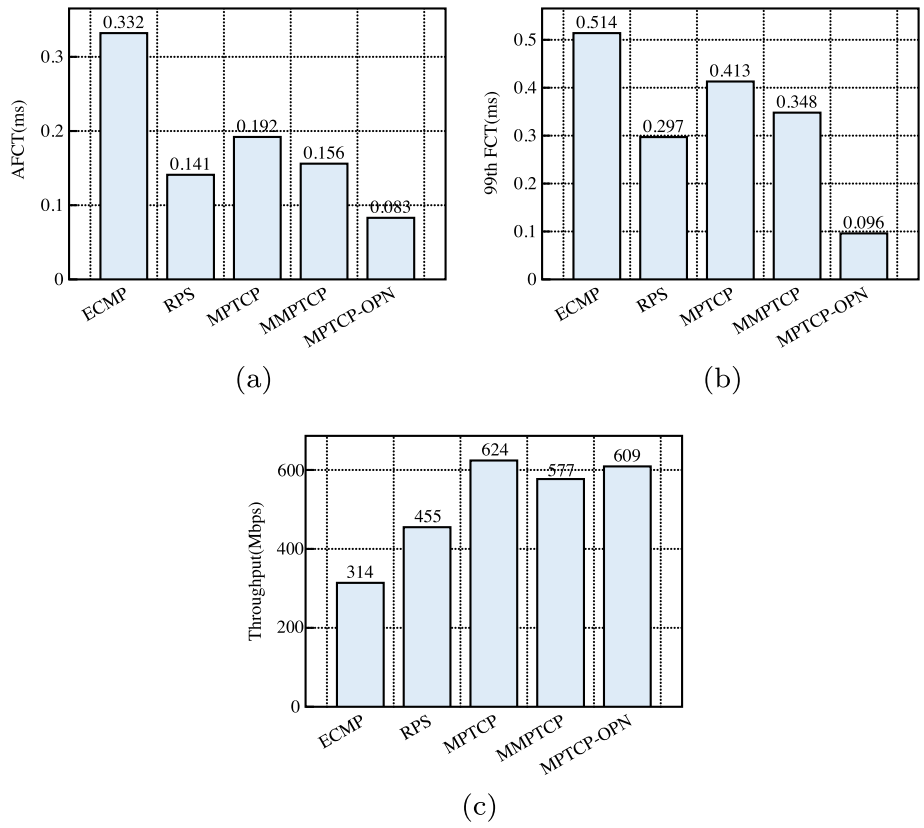

Fig. 11 Performance of heterogeneous flows. **a** AFCT of short flows; **b** 99^{th} percentile FCT of short flows; **c** Throughput of long flows

sizes of 200 long flows and 800 short flows are randomly set in the range of [600, 800] and [40, 60] packets, respectively. The average and 99^{th} percentile flow completion time of short flows, and the throughput of long flows are shown in Fig. 11.

Figure 11a shows the average flow completion time of short flows. Compared with the MPTCP protocol, MPTCP_OPN reduces AFCT by 60%, showing the performance improvement by adjusting the number of subflows. Figure 11b shows the 99^{th} percentile flow completion time of short flows. MPTCP_OPN achieves the best performance in reducing the tail latency, because it avoids the impact of congested path by traffic shifting. Figure 11c shows the throughput of long flows. MPTCP_OPN achieves the similar performance of MPTCP. This result shows that, although MPTCP_OPN reduces the number of subflows, it still obtains good performance of long flows by rapidly transferring traffic and avoiding timeout.

Fairness performance

To evaluate the fairness between MPTCP_OPN and traditional TCP when coexisted in the same network, we employ LIA [17] and OLIA [18] as the congestion control algorithms. In the scenario, the MPTCP_OPN flows and 100 background traditional TCP flows are transmitted on each path. We vary the number of MPTCP_OPN flows and measure the throughputs of MPTCP_OPN and TCP flows on the last hop to the receiver. Figure 12 shows the Jain's fairness index [19] between MPTCP_OPN and traditional TCP flows.

As shown in Fig. 12, the values of Jain's Fairness Indexes between the MPTCP_OPN throughput and the average throughput of background TCP flows are higher than 0.9 under different number of MPTCP_OPN flows. This results show that, when coexist with the traditional TCP flows at the last hop, MPTCP_OPN with the congestion algorithms OLIA and LIA can successfully couple the subflows which share the common bottleneck link, therefore achieving good fairness between MPTCP_OPN and TCP flows.

Performance under realistic workload

Different data center applications have different traffic workloads. To test the performance under realistic workloads, we choose four typical applications, including Data Mining, Web Search, Cache Follower and Web Server. The traffic distributions of four workloads are shown in Table 2 [20]. The experiment generates network traffic

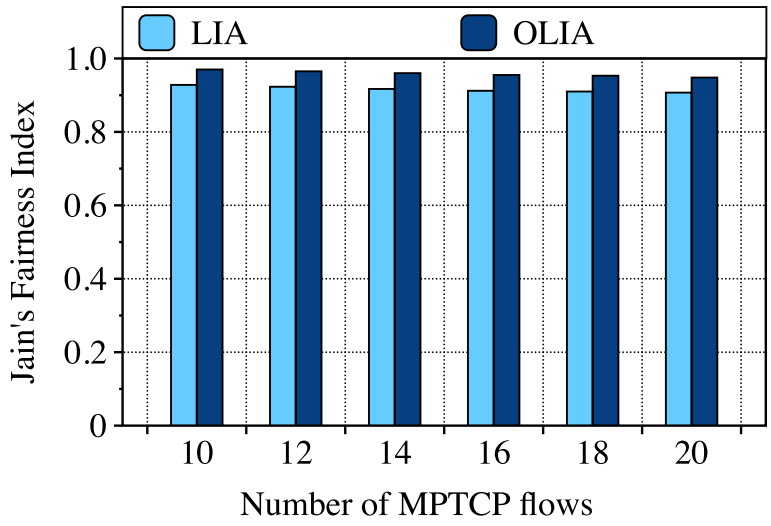

Fig. 12 Jain's fairness index

according to Table 2. The arrival time of realistic workload flows follows the Poisson distribution. Figure 13 shows the experimental results under realistic workloads.

Figure 13a shows the average flow completion time of short flows. Compared with other workloads, the five schemes perform the worst in Data Mining, because the short-flow proportion is the largest in this workload, leading to more full window losses. Although MMPTCP has the advantages of RPS and MPTCP, it does not completely avoid the disadvantages of both: when the network status is not good, it is easy to occur packet reordering problems in RPS phase, and timeout problem in MPTCP phase. Therefore, the performance of MMPTCP is not greatly improved compared with RPS and MPTCP. Overall, MPTCP_OPN effectively avoids timeout events and greatly reduces the average completion time of short flows.

Figure 13b shows the 99^{th} percentile FCT of short flows. ECMP, RPS, MPTCP and MMPTCP all experience severe tail latency, while MPTCP_OPN avoids the impact of timeout events and trailing subflows through traffic shifting and more reasonable subflow cutting, therefore effectively reducing the 99^{th} percentile FCT of short flows. Figure 13c shows the throughput of long flows. For long flows, MPTCP_OPN achieves the similar performance of MPTCP and MMPTCP. Figure 13d shows the overall throughput. MPTCP_OPN effectively avoids timeout events and swiftly shifts the unacknowledged packets to fast paths, thus obtain the highest overall throughput.

Table 2 Flow size distributions of real workload

	Data Mining	Web Search	Cache Follower	Web Server
0-10KB (S)	78%	49%	50%	63%
10KB-100KB (M)	5%	3%	3%	18%
100KB-1MB (L)	8%	18%	18%	19%
1MB- (XL)	9%	20%	29%	-

Performance in asymmetric scenario

The traffic dynamic and switch failure may make the original symmetric network topology into an asymmetric one in production data centers [14]. To MPTCP, link failures and congestions will lead to the degradation of overall throughput, greatly impairing the performance of short flows. The reason is that MPTCP transfers traffic with a fixed number of subflows without considering the link conditions, thus easily suffering from severe throughput degradation from timeout events induced by full window losses, especially for those short flows. Instead, MPTCP_OPN adaptively adjusts the number of subflows according to the real-time network status, and balances traffic based on the dynamic link condition. Therefore, MPTCP_OPN can significantly avoid the timeout events, thus greatly improving the performance of short flows. Besides, the throughput of long flows is not adversely impacted due to its better network adaptability.

In this experiment, we test if MPTCP_OPN is resilient to network asymmetry. Figure 14 shows a typical asymmetric Leaf-Spine topology with 20 available paths between each pair of hosts. However, some links in the dashed lines are not well working, resulting in the uneven bandwidths of 20 paths between the sender and receiver.

First, we test protocol performance in asymmetric network topology. According to the statistics in [15], the maximum number of failed paths is 40% of the total number of paths. We reduce the bandwidth of 8 paths to 100

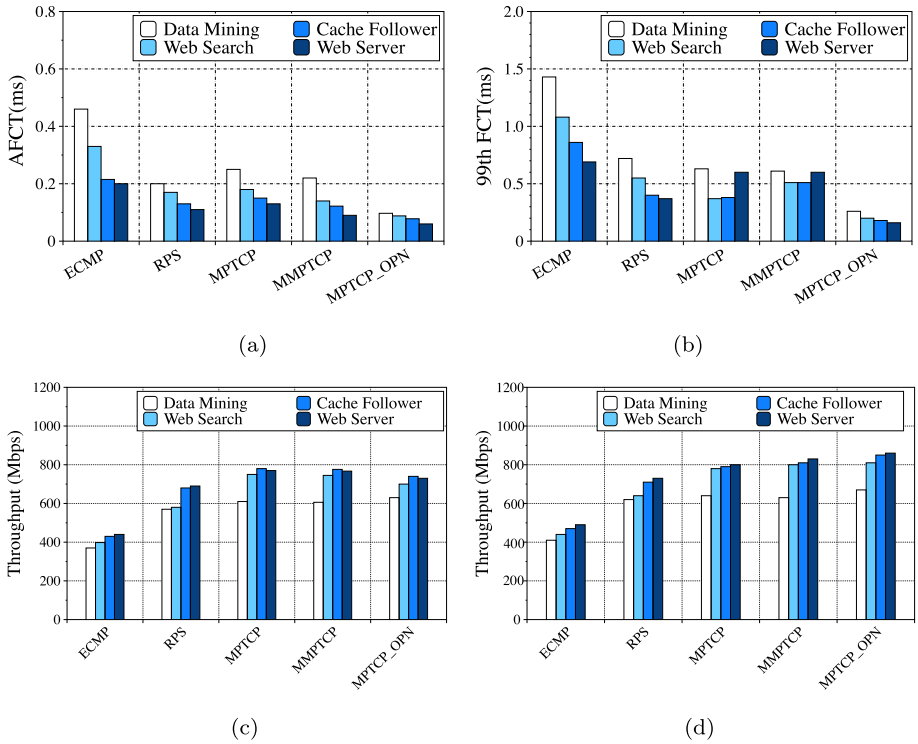

Fig. 13 Performance of realistic workloads. **a** AFCT of short flows; **b** 99^{th} percentile FCT of short flows; **c** Throughput of long flows; **d** Overall Throughput

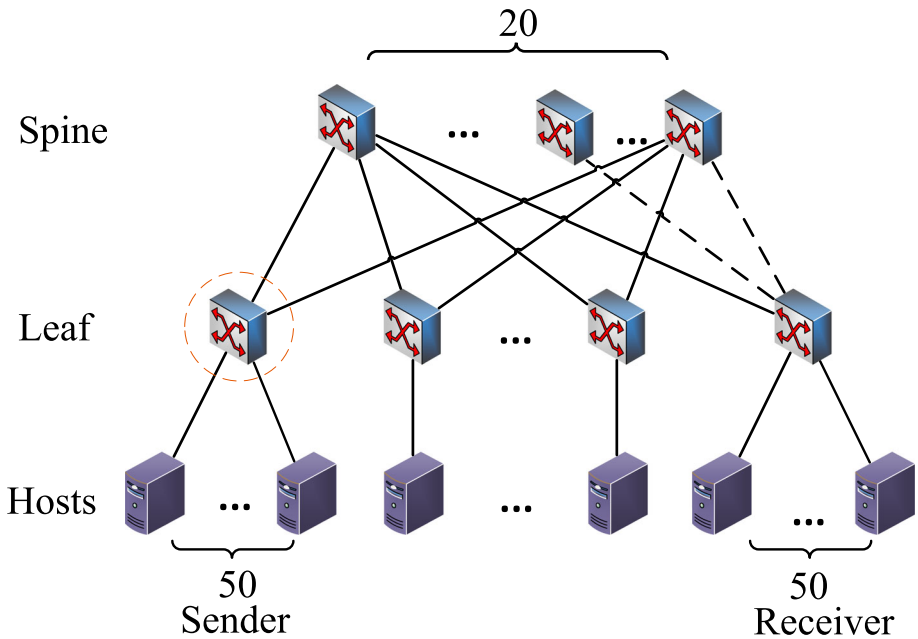

Fig. 14 Asymmetric Leaf-Spine network topology

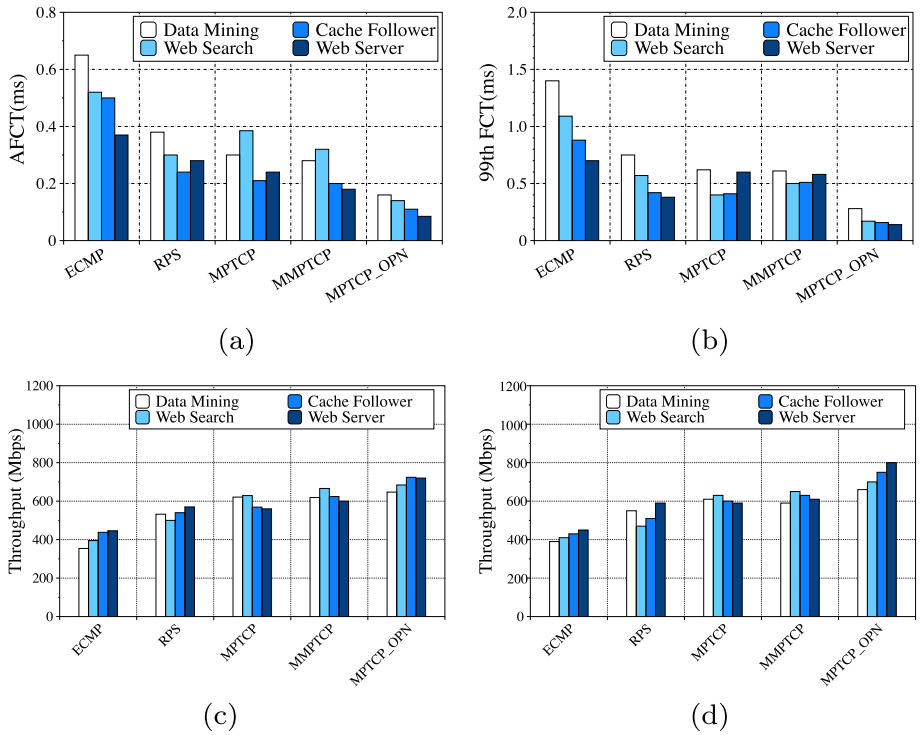

Fig. 15 Performance in asymmetric scenario. **a** AFCT of short flows; **b** 99^{th} percentile FCT of short flows; **c** Throughput of long flows; **d** Overall Throughput

Mbps, and the other 12 paths still have the bandwidth of 1 Gbps. When 50 hosts send data through 20 paths, the ToR switch connected to the sending hosts is overloaded.

The experimental results of protocol performance under asymmetric network topology are shown in Fig. 15. As shown in Fig. 15a and b, under four kinds of workloads, MPTCP_OPN achieves the minimum AFCT and 99^{th} percentile FCT of short flows, because it transfers traffic from the failed path to good ones in asymmetric topology and quickly retransmits the unacknowledged on-flight packets. In Fig. 15c, MPTCP_OPN also achieves the highest throughput of long flows. Since RPS, MPTCP, and MMPTCP transmit data on all paths, the tailing packets and subflows transmitted on the failed path greatly increase the overall flow completion time. Due to its inflexibility, ECMP does not fully utilize the multiple paths. Moreover, when some unlucky flows are transmitted on the failed path by ECMP, the overall transmission is blocked. In Fig. 15d, MPTCP_OPN obtains the maximum overall throughput. MPTCP_OPN adjusts the number of subflows according to network status flexibly and diverts traffic from the failed paths to good ones, thus greatly improve the overall throughput.

Next, we change the bandwidth and number of failed paths to test the average completion time of short flows under different protocols. In the first test, we set the number of failed paths as 8, and increase their bandwidths from 100 Mbps to 1 Gbps. In the second test, the bandwidth of failed path is 100 Mbps, but the number of failed paths is increased from 1 to 8. The experiment generates traffic according to Data Mining and Web Search applications.

Figure 16 shows the protocol performances with failed paths at different bandwidths. When the link bandwidth of failed paths increases, AFCT of short flows gradually decreases. Compared with other schemes, MPTCP_OPN reduces AFCT of short flows by about 50%. Figure 17 shows the protocol performance for different number of failed paths. As the number of failed paths increases, AFCT of short flows gradually increases. In general, since MPTCP_OPN can quickly transfer traffic, it obtains the lowest AFCT of short flows in both application scenarios, showing great resilience to network asymmetry.

Related works

To leverage multipath resources in the multi-rooted tree topologies in data center, many load balancing schemes are proposed in recent years. As the standard load balancing scheme in today's data center, ECMP uses random hashing to assigns flows to different paths, but suffers from the hash collision and under-utilization problem. Freeway [21] adopts different scheduling methods for long

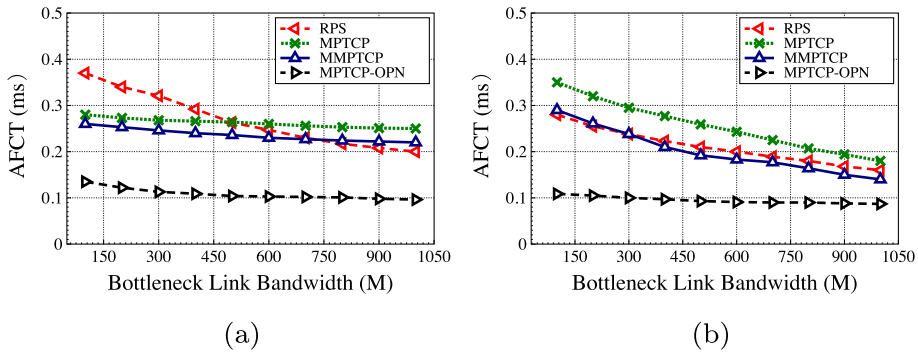

Fig. 16 Performance under different bandwidths of failed paths. **a** Data Mining and **b** Web Search

and short flows in order to meet the different application requirements. Under the condition that the delay-sensitive short flows can be completed within their deadline requirements, the remaining paths are assigned to the long flows.

Unlike ECMP and Freeway using the flow as the switching granularity, RPS randomly sprays each packet onto all available paths to maximize the usage of multipath resources. However, RPS easily leads to packet reordering [22] and unnecessary reduction of transmission rate at the sender. To aviod packet reordering, CAPS [23] encodes the short flows and spreads the packets of short flows to all path. To mitigate the micro-burst at the switch, DRILL [24] picks path for each packet flexibly based on the local queue information. To avoid vigorous rerouting, Hermes [15] reroutes packets in a timely yet cautious manner to good paths only when it will be beneficial.

Compared with the flow-based and packet-based load balancing designs, MPTCP transmits data in unit of subflow to make good use of multipath resources and meanwhile avoid packet reordering in subflows. To improve the MPTCP performance in data center networks, a rich body of work [2, 6, 25–29] has thus emerged. In order to understand the performance of MPTCP protocol better, the literature [30, 31] tests the influence of different topologies and traffic distributions on MPTCP. The experimental results show that MPTCP protocol can effectively utilize the available bandwidth to provide higher throughput, better fairness and stronger robustness than TCP protocol.

In order to assign each subflow of MPTCP on different idle paths, Dual-NAT [26] uses NAT technology to dynamically construct disjoint paths for each subflow of MPTCP, thus making full use of the path diversity of data center network and improve total throughput. FUSO [32] improves the retransmission mechanism of MPTCP subflow. When the sender suspects that a packet is lost, it will use the remaining congestion windows of other less congested subflows to quickly retransmit the lost data packets, thereby avoiding the influence of the trailing subflows and reducing the total completion time of subflows.

MPTCP achieves full link utilization by exhausting the link buffer, thus causing considerable queueing delay and packet loss. This affects the performance of the delay-sensitive short flows under the overwhelm data of throughput-sensitive long flows. To solve this problem, XMP [33] uses the ECN mechanism to control the occupancy of switch buffer and reduce the impact of long flows on short ones. MMPTCP [34] distinguishes the long and short flows according to the number of bytes that

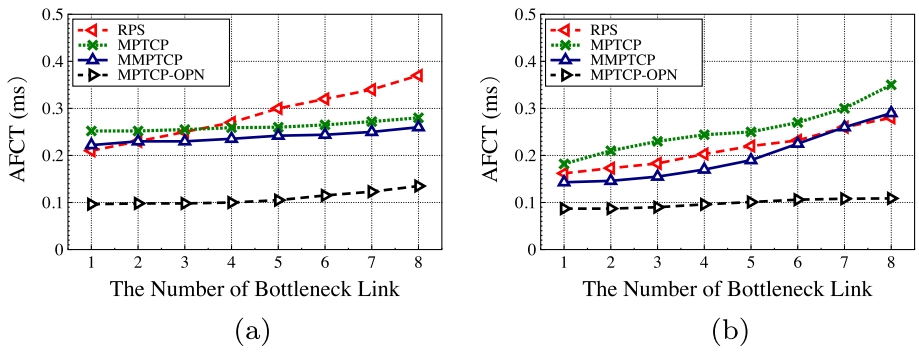

Fig. 17 Performance under different number of failed paths. **a** Data Mining and **b** Web Search

have been sent, and adopts different transmission strategies for long and short flows. For the short flows, RPS is used to quickly utilize available path resources and reduce flow completion time. For the long flows, MPTCP subflow transmission strategy is still used to ensure high throughput. To avoid packet reordering at a receiver, Promenade [35] utilizes random network coding. With random network coding, packets belonging to the same flow are split into multiple subflows, and then forwarded along different paths. With the prevalent of RDMA, literature [36] proposes a multi-path transport for RDMA named MP-RDMA. MP-RDMA employs novel multi-path ACK-clocking and out-of-order aware path selection to choose the best paths and forward packets in a congestion-aware fashion.

In order to avoid the asymmetry problem caused by the latency difference, STMS [37] scheduler sends packets with smaller sequence number to a fast path while sending packets with larger sequence number to a slow path, allowing packets sent over different paths to arrive simultaneously. Considering the continuously increasing demand for mobile communication, literature [38] presents two novel scheduling schemes which are the block estimation (BLEST) scheduler and the shortest transmission time first (STTF) scheduler to guarantee low-latency communication of MPTCP. To improve the performance of latency-sensitive applications over MPTCP in high-delay and lossy networks, literature [39] proposes a new framework using a XOR-based dynamic FEC scheme to reduce the flow completion time.

To improve the performance of MPTCP for rack-local or many-to-one traffic, DCMPTCP [40] recognizes rack-local traffic and eliminates unnecessary subflows to reduce the overhead. Then it estimates the length of flow and establishes subflows in an intelligent way. In addition, DCMPTCP enhances explicit congestion notification to improve the performance of congestion control on inter-rack many-to-one short flows.

In order to reduce the energy consumption of multipath transmission, MPTCP_D [41] checks whether multiple subflows share a common link. In order to prevent multiple subflows from being transmitted on the same path and reduce energy consumption, the literature [42] establishes the MPTCP energy consumption model and shifts the traffic to paths with low energy consumption.

MPTCP protocol and its enhanced mechanisms use all available paths to speed up data transmission, while ignoring the negative impact of excessive number of subflows on transmission efficiency. In fact, too many subflows shrink flow size of each subflow, resulting in large timeout probability and low transmission efficiency. Compared with these schemes, our design MPTCP_OPN adjusts the number of subflows and uses traffic shifting to achieve the good tradeoff between link utilization and tail latency under network dynamic.

Conclusion

This work presents MPTCP_OPN for highly dynamic traffic in data center networks. We adjust the number of subflows to mitigate the full window loss according to network status. Moreover, we design the traffic shifting to quickly retransmit the blocked packets on congested paths. MPTCP_OPN is deployed only at the sender side, which avoids modifying switches. To test MPTCP_OPN's broad applicability and effectiveness, we evaluate our design on large-scale simulation test under realistic traffic workloads. The results indicate that with MPTCP_OPN, we remarkably reduce the flow completion time by up to 50% compared with the state-of-the-art load balancing designs, such as RPS, MPTCP and MMPTCP.

Acknowledgements
We would like to thank Dr. Sen Liu for his valuable help in evaluation.

Authors' contributions
All authors have participated in conception and design, or analysis and interpretation of this paper.

Funding
This work is supported by National Natural Science Foundation of China (61872387, 61872403) and CERNET Innovation Project (NGII20170107).

Availability of data and materials
Not applicable.

Competing interests
The authors declare that they have no competing interests.

Author details
[1]School of Computer Science and Engineering, Central South University, 410083 Changsha, China. [2]Hunan Provincial Key Laboratory of Intelligent Computing and Language Information Processing, Hunan Normal University, 410081 Changsha, China.

Received: 26 October 2019 Accepted: 10 February 2020
Published online: 24 February 2020

References

1. Al-Fares M, Loukissas A, Vahdat A. A scalable, commodity data center network architecture. In: Proceedings of the ACM SIGCOMM 2008 conference on Data communication - SIGCOMM '08. ACM Press; 2008. https://doi.org/10.1145/1402958.1402967.
2. Benson T, Akella A, Maltz DA. Network traffic characteristics of data centers in the wild. In: Proceedings of the 10th annual conference on Internet measurement - IMC '10. ACM Press; 2010. https://doi.org/10.1145/1879141.1879175.
3. Hopps C. Analysis of an equal-cost multi-path algorithm, RFC2992. 2000. https://doi.org/10.17487/RFC2992. Accessed 01 Nov 2000.
4. Dixit A, Prakash P, Hu YC, Kompella RR. On the impact of packet spraying in data center networks. In: 2013 Proceedings IEEE INFOCOM. IEEE; 2013. https://doi.org/10.1109/infcom.2013.6567015.
5. Ford A, Raiciu C, Handley M, Bonaventure O. Tcp extension for multipath operation with multiple addresses, RFC6824. 2013. https://doi.org/10.17487/RFC6824. Accessed 18 Jan 2013.
6. Chen Y, Griffifh R, Liu J, Katz RH, Joseph AD. Understanding TCP incast throughput collapse in datacenter networks. In: Proceedings of the 1st ACM workshop on Research on enterprise networking - WREN '09. ACM Press; 2009. https://doi.org/10.1145/1592681.1592693.

7. Ren Y, Zhao Y, Liu P, Dou K, Li J. A survey on tcp incast in data center networks. Int J Commun Syst. 2014;27:1160–72.
8. Qin Y, Yang W, Ye Y, Shi Y. Analysis for tcp in data center networks: Outcast and incast. J Netw Comput Appl. 2016;68:140–50.
9. Shan D, Ren F. Ecn marking with micro-burst traffic: Problem, analysis, and improvement. IEEE/ACM Trans Netw. 2018;26:1533–46.
10. Huang J, Li S, Han R, Wang J. Receiver-driven fair congestion control for tcp outcast in data center networks. J Netw Comput Appl. 2019;131: 75–88.
11. Multipath TCP Implementation in the Linux Kernel. http://www.multipath-tcp.org. Accessed 04 June 2017.
12. Liu S, Huang J, Zhou Y, Wang J, He T. Task-aware tcp in data center networks. IEEE/ACM Trans Netw. 2019;27:389–404.
13. Li Y, Miao R, Kim C, Yu M. Lossradar: Fast detection of lost packets in data center networks. In: Proceedings of the 12th ACM Conference on Emerging Networking Experiments and Technologies - CoNEXT '16. New York: ACM Press; 2016. p. 1–15. https://doi.org/10.1145/2999572.2999609.
14. Gill P, Jain N, Nagappan N. Understanding network failures in data centers: Measurement, analysis and implications. In: Proceedings of the Conference on Special Interest Group on Data Communication - SIGCOMM '11. New York: ACM Press; 2011. p. 350–361. https://doi.org/10.1145/2018436.2018477.
15. Zhang H, Zhang J, Bai W, Chen K, Chowdhury M. Resilient datacenter load balancing in the wild. In: Proceedings of the Conference of the ACM Special Interest Group on Data Communication - SIGCOMM '17. ACM Press; 2017. https://doi.org/10.1145/3098822.3098841.
16. Alizadeh M, Greenberg A, Maltz D, Padhye J, Patel P, Prabhakar B, Sengupta S, Sridharan M. Data center tcp (dctcp). In: Proceedings of the ACM SIGCOMM 2010 conference on SIGCOMM - SIGCOMM '10. ACM Press; 2010. https://doi.org/10.1145/1851182.1851192.
17. Raiciu C, Handley M, Wischik D. Coupled congestion control for multipath transport protocols, RFC 6356. 2011. https://rfc-editor.org/rfc/rfc6356.txt. Accessed 14 Oct 2011.
18. Khalili R, Gast N, Popovic M, Boudec J. Mptcp is not pareto-optimal: Performance issues and a possible solution. IEEE/ACM Trans Netw. 2013;21:1651–65.
19. Jain R, Chiu D, Hawe W. A quantitative measure of fairness and discrimination for resource allocation in shared computer system. Digital Equipment Corporation; 1984. https://arxiv.org/abs/cs/9809099.
20. Cho I, Jang K, Han D. Credit-scheduled delay-bounded congestion control for datacenters. In: Proceedings of the Conference of the ACM Special Interest Group on Data Communication - SIGCOMM '17. ACM Press; 2017. p. 239–252. https://doi.org/10.1145/3098822.3098840.
21. Wang W, Sun Y, Salamatian K, Li Z. Adaptive path isolation for elephant and mice flows by exploiting path diversity in datacenters. IEEE Trans Netw Serv Manag. 2016;13:5–18.
22. He K, Rozner E, Agarwal K, Felter W, Carter J, Akella A. Presto: Edge-based load balancing for fast datacenter networks. In: Proceedings of the Conference on Special Interest Group on Data Communication - SIGCOMM '15. New York: ACM Press; 2015. https://doi.org/10.1145/2785956.2787507.
23. Hu J, Huang J, Lv W, Zhou Y, Wang J. Caps:coding-based adaptive packet spraying to reduce flow completion time in data center. IEEE/ACM Trans Netw. 2019;27:2338–53.
24. Ghorbani S, Yang Z, Godfrey PB, Ganjali Y, Firoozshahian A. Drill: Micro load balancing for low-latency data center networks. In: Proceedings of the Conference on Special Interest Group on Data Communication - SIGCOMM '17. New York: ACM Press; 2017. https://doi.org/10.1145/3098822.3098839.
25. Kandula S, Sengupta S, Greenberg A, Patel P, Chaiken R. The nature of data center traffic: Measurements and analysis. In: Proceedings of the 9th Annual Conference on Internet Measurement - IMC '09. New York: ACM Press; 2009. https://doi.org/10.1145/1644893.1644918.
26. Cao Y, Xu M. Dual-nat: Dynamic multipath flow scheduling for data center networks. In: Proceedings of the IEEE ICNP: 7-10 October 2013; Goettingen. IEEE; 2013. https://doi.org/10.1109/icnp.2013.6733629.
27. Agache A, Deaconescu R, Raiciu C. Increasing datacenter network utilisation with grin. In: Proceedings of the 12th USENIX Conference on Networked Systems Design and Implementation - NSDI '2015. Berkeley: USENIX Association Press; 2015. https://www.usenix.org/conference/nsdi15/technical-sessions/presentation/agache.
28. Habib S, Qadir J, Ali A, Habib D, Li M, Sathiaseelan A. The past, present, and future of transport-layer multipath. J Netw Comput Appl. 2016;75: 236–58.
29. Huang J, Huang Y, Wang J, He T. Packet slicing for highly concurrent tcps in data center networks with cots switches. IEEE Trans Cloud Comput. https://doi.org/10.1109/TCC.2018.2810870.
30. Raiciu C, Barre S, Pluntke C, Greenhalgh A, Wischik D, Handley M. Improving datacenter performance and robustness with multipath tcp. In: Proceedings of the ACM SIGCOMM 2011 conference on SIGCOMM - SIGCOMM '11. ACM Press; 2011. https://doi.org/10.1145/2018436.2018467.
31. Wischik D, Raiciu C, Greenhalgh A. M H. Design, implementation and evaluation of congestion control for multipath tcp. In: Proceedings of the 8th USENIX Conference on Networked Systems Design and Implementation - NSDI '2011. Berkeley: USENIX Association Press; 2011. p. 99–112. https://www.usenix.org/legacy/event/nsdi11/tech/full_papers/Wischik.pdf.
32. Chen G, Lu Y, Meng Y, Li B, Tan K, Pei D, Cheng P, Luo L, Xiong Y, Wang X, Zhao Y. Fuso: Fast multi-path loss recovery for data center networks. IEEE/ACM Trans Netw. 2018;26:1376–89.
33. Cao Y, Xu M, Fu X, Dong E. Explicit multipath congestion control for data center networks. In: Proceedings of the ninth ACM conference on Emerging networking experiments and technologies - CoNEXT '13. ACM Press; 2013. https://doi.org/10.1145/2535372.2535384.
34. Kheirkhah M, Wakeman I, Parisis G. Multipath transport and packet spraying for efficient data delivery in data centres. Comput Netw. 2019;162:1–15.
35. Chen L, Feng Y, Li B, Li B. Promenade: Proportionally fair multipath rate control in datacenter networks with random network coding. IEEE Trans Parallel Distrib Syst. 2019;30:2536–46.
36. Lu Y, Chen G, Li B, Xiong Y, Cheng P, Zhang J, Chen E, Moscibroda T. Multi-path transport for rdma in datacenters. In: Proceedings of the 15th USENIX Conference on Networked Systems Design and Implementation - NSDI '2018. Berkeley: USENIX Association Press; 2018. p. 357–371. https://www.usenix.org/conference/nsdi18/presentation/lu.
37. Shi H, Cui Y, Wang X, Hu Y, Dai M, Wang F, Zheng K. Stms: Improving mptcp throughput under heterogeneous networks. In: Proceedings of the USENIX Annual Technical Conference - ATC '2018. Berkeley: USENIX Association Press; 2018. p. 719–730. https://www.usenix.org/conference/atc18/presentation/shi.
38. Hurtig P, Grinnemo K, Brunstrom A, Ferlin S, Alay O, Kuhn N. Low-latency scheduling in mptcp. IEEE/ACM Trans Netw. 2019;27:302–15.
39. Ferlin S, Kucera S, Claussen H, Alay O. Mptcp meets fec: Supporting latency-sensitive applications over heterogeneous networks. IEEE/ACM Trans Netw. 2018;26:2005–18.
40. Dong E, Fu X, Xu M, Yang Y. Dcmptcp: Host-based load balancing for datacenters. In: Proceedings of the IEEE ICDCS: 2-6 July 2018; Vienna. IEEE; 2018. https://doi.org/10.1109/icdcs.2018.00067.
41. Zhao J, Liu J, Wang H, Xu C. Multipath tcp for datacenters: From energy efficiency perspective. In: IEEE INFOCOM 2017 - IEEE Conference on Computer Communications. IEEE; 2017. https://doi.org/10.1109/infocom.2017.8057182.
42. Zhao J, Liu J, Wang H. On energy-efficient congestion control for multipath tcp. In: Proceedings of the IEEE ICDCS: 5-8 June 2017; Atlanta. IEEE; 2017. https://doi.org/10.1109/icdcs.2017.156.

Publisher's Note