Journal of Network and Computer Applications 141 (2019) 73–85

Contents lists available at ScienceDirect

Journal of Network and Computer Applications

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

Improve MPTCP with SDN: From the perspective of resource pooling

Yanbing Liu, Xiaowei Qin *, Ting Zhu, Xiaohui Chen, Guo Wei

CAS Key Laboratory of Wireless-Optical Communications, University of Science and Technology of China, Hefei 230027, China

ARTICLE INFO

Keywords:
Multipath TCP
Software defined network
Resource pooling

ABSTRACT

Multipath TCP (MPTCP) is proposed by IETF to support concurrent multipath transmission between multihomed hosts. Resource pooling principle, which treats network as a single pooled resource, is applied in the design of MPTCP schemes, and provides guidelines for MPTCP's control to achieve resource utilization and fair resource allocation. However, randomized routing solution and load balancing without global network information in MPTCP have become obstacles to achieve these objectives, which is validated through our measurements. To address above problems, we propose S-MPTCP, which realizes coupling control to MPTCP connections in network based on SDN technology providing global network information. Leveraging collected topology information, a key parameter, expected throughput, is calculated for each subflow ensuring the efficiency and fairness in resource exploration and allocation, and according to it, routing and load balancing modules are improved. Under S-MPTCP's control, subflow's throughput will stabilize around its expected throughput, and meanwhile congestion can be alleviated effectively. Experimental results show that S-MPTCP achieves significant enhancement of bandwidth utilization and fair resource allocation, and the time needed for load balancing is shortened considerably.

1. Introduction

Nowadays, network develops on a trend toward multipath and most smart terminals have been equipped with multiple network interfaces. Regular Transmission Control Protocol (TCP), which is designed to be a single-path protocol in essence, cannot provide management for multipath transmission. On mobile devices, when Wi-Fi/4G handover happens, existing TCP connections must be torn down before new connections are established and this service interruption will tarnish user experience; Although there are often multiple available paths between two hosts in modern data centers, regular TCP protocol can only choose one link randomly and a great deal of network resources are wasted.

To fully utilize network resource between multihomed hosts, Multipath TCP (MPTCP) (Ford et al., 2013) is proposed by the Internet Engineering Task Force (IETF), which allows a single connection to transmit packets on multiple available paths simultaneously. MPTCP has been applied to kinds of use cases: Cooperation of Wi-Fi and 4G improves transmission performance on mobile phones with MPTCP (Paasch et al., 2012); In data centers, multiple paths bandwidth between two servers can be aggregated efficiently (Raiciu et al., 2011a).

Due to the multipath characteristic of MPTCP, a novel architectural principle, resource pooling principle (Wischik et al., 2008), is applied in the design of MPTCP schemes. In the concept of resource pooling principle, network is treated as a single pooled resource, and it should be allocated to every connection in an efficient and fair manner, to ensure reliable and high-speed transmission on all connections. Through a series of specifically designed modules including path management and congestion control, resource pooling is achieved in theory.

However, MPTCP practical control effect is barely satisfactory according to our experiments results introduced in Section 2. The defects on control mechanism design impede stable, responsive and fair control. The path selection scheme of MPTCP is usually combined with randomized load balancing technologies such as Equal-cost multi-path Routing (ECMP) (Hopps, 2000). A limitation of this random path selection is that avoidable bottlenecks may be created and mass network resource is idle, and meanwhile the throughput gap between connections can be tremendous. There are also flaws in MPTCP's congestion control module. In the Slow Start phase, the initial value of ssthresh is set to infinity in MPTCP stack according to RFC5681 (Allman et al., 2009), which should be set to a more reasonable value. Loss-based algorithms are adopted by MPTCP as the default congestion control solu-

* Corresponding author.
E-mail addresses: viper@mail.ustc.edu.cn (Y. Liu), qinxw@ustc.edu.cn (X. Qin), zhuting@mail.ustc.edu.cn (T. Zhu), cxh@ustc.edu.cn (X. Chen), wei@ustc.edu.cn (G. Wei).

https://doi.org/10.1016/j.jnca.2019.05.015
Received 24 July 2018; Received in revised form 19 March 2019; Accepted 26 May 2019
Available online 30 May 2019

tion, which means that load balancing depends on the event of packet loss. This coarse-grained and reactive load balancing method also hurts the performance of MPTCP, and there is often a significant difference between actual load balance result and ideal result. Although the network bandwidth is usually fully utilized, the resource allocation cannot be guaranteed to be performed fairly, and the time required for load balancing is unfriendly for small and medium-size flows. Therefore, it's essential to refine control mechanism of MPTCP to address these problems and achieve desirable resource pooling control.

Software Defined Network (SDN), which is considered as a compelling technology in the future network, is a potential solution for the proceeding problems of MPTCP. One of the major advantages of SDN technology is that the centralized controller has visibility of the complete network, the current traffic characteristics and the load on individual network elements. It is thus in a prime position to explore, allocate and manage network resource for each connection, which can play an important role in achieving the resource pooling.

In this paper, we propose SDN-based MPTCP (S-MPTCP), which provides a holistic solution to correct above-mentioned defects of MPTCP and finally achieve the control effect of resource pooling based on the cooperation of SDN controller and the MPTCP protocol stack. The system design can be divided into three parts: The first phase is to explore available bandwidth resource in network according to network statistics collected by SDN controller, and output optimal subflows routes set for newly-established MPTCP connection. Next step is the allocation of resource explored in the first stage, and a key parameter, expected throughput, is calculated and sent to corresponding host for all related subflows in the connection set. The first two steps are performed in the SDN controller, and the final part is to realize efficient and fair rate control in the MPTCP stack at the side of host leveraging the information transmitted from the SDN controller. According to expected throughput parsed from the MPTCP option, the comparison result of subflow's current throughput and its expected throughput will be a significant indicator, which provides guidance the congestion window control in both slow start phase and congestion avoidance phase. Our control modules in MPTCP stack look for the control effect that, subflow's transmission rate will stabilize around expected throughput, and congestion will be prevented in its incipient stage. Under our control mechanism, network bandwidth resource in the resource pool is fully explored and allocated to connections fairly in a short time, and meanwhile congestion at the bottleneck link is restrained efficiently.

The main contributions of this work are as follows:

- We analyze the existing issues in MPTCP control mechanism based on experiments result. Lack of global network information leads to several significant design defects, including randomized subflow routes selection and coarse-grained, reactive load balancing, and practical control effect of current MPTCP stack is far from resource pooling demands.
- To address these problems, we propose S-MPTCP, a system based on the cooperation of SDN and MPTCP technologies, which aims to realize efficient resource exploration and fair resource allocation for connections in network. In the control layer, SDN controller will be responsible for resource exploration and allocation in network, which includes the routing assignments and expected throughput calculation for subflows; In the infrastructure layer, MPTCP stack will process the information from SDN controller, and perform path management and rate control accordingly. Several related algorithms are designed to accomplish above functions.
- The detailed architecture of S-MPTCP is designed and we implement it based on MPTCP Linux kernel, Mininet and Floodlight controller, and experiments are performed to evaluate the performance of our system. Experiment results demonstrate that S-MPTCP can maximize total system bandwidth and prevent avoidable flow collision through subflow route assignment in SDN controller. Meanwhile fairness and responsiveness are improved significantly in the process of resource allocation compared with regular MPTCP.

The remainder of this paper is structured as follows. Section 2 introduces the motivation of our work, including the analyze of issues in MPTCP and introduction of related work. In Section 3, we present the design of S-MPTCP and algorithms. Section 4 focuses on the architecture of S-MPTCP system. In Section 5, the performance of S-MPTCP is evaluated. Section 6 concludes the paper and discusses the issues about cooperation of MPTCP and SDN in the future investigation.

2. Motivation

In this section, we will first introduce resource pooling principle and methods in MPTCP to achieve resource pooling control. Then experiments are performed to examine whether MPTCP approaches the ideal control effect of resource pooling. In the end of this section, we introduce and analyze several existing researches which use SDN to improve MPTCP's performance.

2.1. Background

The most important difference between MPTCP and traditional single-path TCP is that the transmission of a single connection can be performed on multiple paths simultaneously. In RFC6824 (Ford et al., 2013), general MPTCP operations are defined, including connection and subflow initiation, data sequence mapping, address information exchange, etc. These operations provide architecture basis for multipath transmission.

Apart from these basic structural operations, there are some other significant issues on MPTCP flow control. The schemes of path selection, congestion control adopted by MPTCP stack decide the performance of transmission. In (Wischik et al., 2008), resource pooling principle is proposed for multipath-capable end systems. The resource in network is treated as a whole called resource pool. Resource in the pool should be well utilized and allocated to connections in a fair manner illustrated in Fig. 1 through routing and load balancing, and ultimately the efficiency and fairness of network are ensured.

In the design of MPTCP, congestion control scheme is the centralized embodiment of resource pooling principle. Based on resource pooling principle, Raiciu et al. (2011b) proposed three major design goals of MPTCP congestion control mechanism: improve throughput,

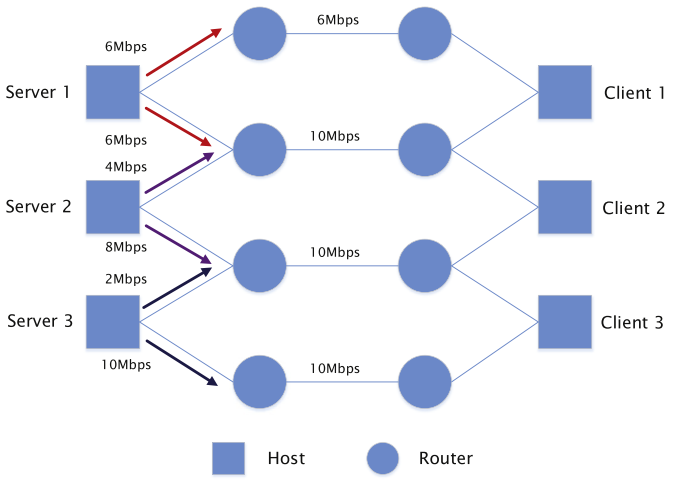
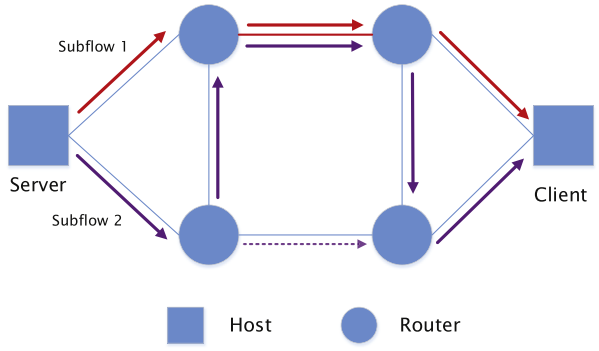


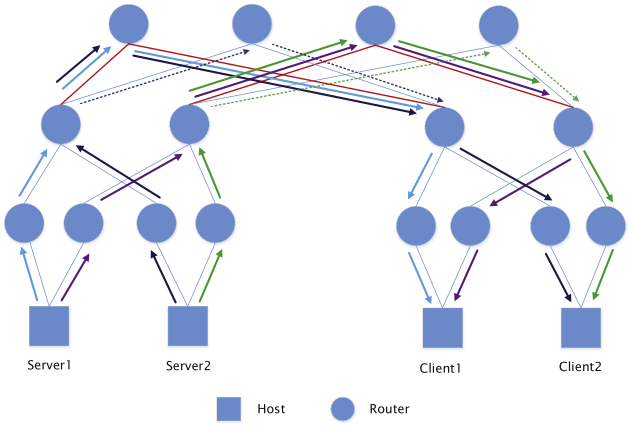
Fig. 1. Fair resource allocation. The entire 36 Mbps bandwidth is allocated for MPTCP connections equally. Arrows in different colors represent data transfer directions of different MPTCP connections. (For interpretation of the references to color in this figure legend, the reader is referred to the Web version of this article.)

do no harm and balance congestion. In Goal 1, MPTCP is expected to achieve higher resource utilization than single path TCP (SPTCP). Goals 2 ensures fairness of the resource pool, and resource should be shared by connections fairly. Goal 3 also captures the concept of resource pooling: If each multipath flow sends more data through its least congested path, the traffic in the network will move away from congested areas. To summarize, the congestion control of MPTCP is performed under the guidance of resource pooling principle to achieve desirable balance of efficiency and fairness through coupled cwnd control using network statistics from local host. Several coupled congestion control algorithms including LIA (Wischik et al., 2011), OLIA (Khalili et al., 2013), BALIA (Peng et al., 2016) and wVegas (Cao et al., 2012) have been proposed and implemented for MPTCP. LIA, OLIA and BALIA are loss-based congestion control algorithms, while wVegas is delay-based which is originated from TCP Vegas (Brakmo and Peterson, 1995).

The current routing technology adopted by MPTCP is ECMP which uses random hashing to distribute subflows. This randomized method may lead to two kinds of avoidable flow collisions: collisions between subflows belonging to the same connection and collisions between subflows belonging to different connections. Fig. 2 illustrates these two kinds of collisions respectively. Such collisions will result in the insufficient utilization of network resource and increase the probability of congestion.



(a) Collision between subflows belonging to the same MPTCP connection



(b) Collisions between subflows belonging to different MPTCP connections

Fig. 2. Two kinds of flow collisions due to randomized path selection. Arrows in different colors represent data transfer directions of different MPTCP connections. Red lines represent links where collisions happen on, and dotted lines represent ideal paths which can avoid collisions. (For interpretation of the references to color in this figure legend, the reader is referred to the Web version of this article.)

Table 1
Connection measurement result.

Algorithm	Total Throughput/Mbps	SD/Mbps	Convergence Time/s
BALIA	33.962	3.511	13.8
OLIA	33.950	3.350	11.8
LIA	33.967	2.847	11.0
wVegas	33.123	0.992	6.2
Ideal Value	36	0	0

2.2. Evaluation of MPTCP performance

Through practical experiments, we investigate the performance of MPTCP and check its efficiency, fairness and responsiveness in networks which contain multiple connections.

First, we exclude the influence of routing and only examine the fairness and responsiveness achieved by MPTCP congestion control. The topology and network parameter is shown in Fig. 1. Three MPTCP connections 1–3 with two subflows share the entire 36 Mbps bandwidth resource. All other links' bandwidths are set to 10 Mbps and all links' delays are 5 ms. We use Mininet to create the topology and 50s long flows are generated for each connection using Iperf with a 5s interval between connection establishments. Total throughput, standard deviation (SD) between connections' throughputs and convergence time are calculated to evaluate efficiency, fairness and responsiveness. Total throughput and standard deviation are calculated for statistics after load balancing has ended and throughputs of all connections have stabilized, and the convergence time is defined as the consuming time of load balancing before all connections' throughputs stabilize. Experiment result is shown in Table 1.

From these experiment results, there exists an obvious disparity between practical effect of MPTCP loss-based congestion control and ideal control effect in Fig. 1. Although the bandwidth utilization is high, the fairness and responsiveness are not guaranteed. Standard deviation statistics suggest that there are obvious throughput gaps between connections. The most common behavior with all three algorithms is shown in Fig. 3, where we can observe that throughput of connection 3 approaches the sum of throughputs of connection 1 and connection 2. Therefore, fairness in resource allocation is not fully achieved by these algorithms and a considerable portion of expected bandwidth of some connections is occupied by other connections due to defects in MPTCP congestion control. Another significant issue is responsiveness. Results show that it will take more than 10 s to complete load balancing before throughputs of connections converge, which means the efficiency and fairness provided by MPTCP control will be very limited for short and medium flows. Therefore, MPTCP loss-based congestion control algorithms perform poorly on both fairness and responsiveness.

Delay-based congestion control algorithm turns in a better performance in above experiments compared with loss-based algorithms,

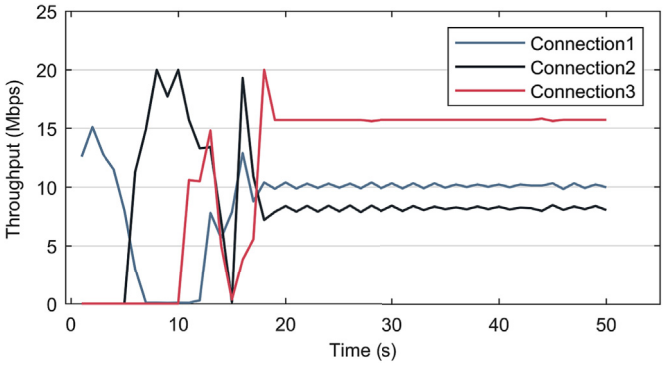


Fig. 3. Throughput trajectories of competing connections with BALIA.

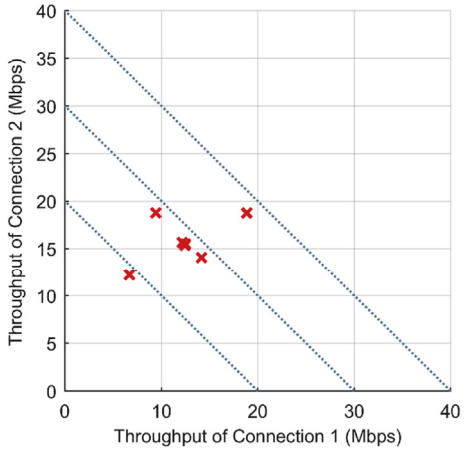


Fig. 4. Throughputs of connections in the routing solution experiment.

but is also not so satisfactory. The fairness of resource allocation is improved, but these is still an about 2 Mbps throughput gap between connections. Similarly, the time required for load balancing is not friendly to short and medium flows.

Next, the bandwidth utilization under ECMP routing solution of MPTCP is investigated on the topology shown in Fig. 2(b). All links are set to with 10 Mbps bandwidth and 5 ms delay. Connection 1 and Connection 2 share the entire 40 Mbps bandwidth resource in the network. Experiment is repeated for 10 times, and results are shown in Fig. 4. From Fig. 4, we can observe that the total throughputs of connections can be divided into three levels: 40 Mbps, 30 Mbps and 20 Mbps. According to topology information, the total throughput will be limited to 30 Mbps when a collision happens on one path and be limited to 20 Mbps when collisions happen on both paths, which is consistent with the experimental result. This experiment makes clear that randomized routing selection adopted by MPTCP cannot fully explore and utilize the resource in the network, which results in performance degradation of connections and waste of network resource.

Our experiments indicate that MPTCP performs poorly both on resource exploration and resource allocation due to a series of flaws in congestion control and routing modules. It's essential to correct these defects and perfect control mechanism for further performance improvement of MPTCP.

2.3. Related works

As the key technology in next generation network, SDN increasingly attracts attention in scientific and industrial communities. Several researches have been proposed to evaluate and optimize TCP with SDN. In (Lai et al., 2019), two analytical models are developed for evaluating the performance of TCP and UDP flows over SDN. The simulation results show that TCP significantly outperforms UDP in SDN networks. Congestion control model of TCP is improved in (Bao et al., 2018; Abdelmoniem et al., 2017; Singh et al., 2019). A centralized congestion avoidance mechanism using SDN technology is proposed in (Bao et al., 2018), which can solve bottleneck change and burst traffic problems while ensuring fairness. In data centers networks, a SDN-based congestion control framework is designed in (Abdelmoniem et al., 2017) to reduce the completion time of short-lived incast flows. In wireless networks (Singh et al., 2019), uses SDN controller to detect movement and hide mobility induced packet losses, which can avoid unwanted reduction in sending rate.

Various researchers have proposed improvement schemes for several modules in MPTCP using SDN technology, although there is still no related work to improve congestion control module of MPTCP which is vital in the control mechanism of MPTCP till now. An important scheme in MPTCP is path management which is responsible for sharing address information and management of subflows. Path heterogeneity between subflows will degrade MPTCP's performance significantly (Yedugundla et al., 2016; Cordero, 2016; Ramaboli et al., 2012). In (Nam et al., 2016), subflows' path capacities are measured and the gap between capacities is calculated. Poor subflows which provide a low capacity will be removed form the list of available subflows to avoid the increase of reordering queue length in receive buffer. In (Kukreja et al., 2016), researchers implemented an SDN-based automated testbed to inspect the influence of delay and delay differences to MPTCP performance. Both high delay and high delay differences lead to severe performance degradation according to their measurement, and this result can become a reference of path management module design. Besides, in (Chattopadhyay et al., 2018), researchers propose a SDN aided method to predict the aggregated throughput of a MPTCP flow based on path characteristics, and find out the optimal set of subflows.

Routing for subflows is another research aspect in the spotlight. In (Sandri et al., 2015), a shortest path is selected for the first subflow of connection via Dijkstra algorithm and disjointed paths are selected for subsequent subflows of the same connection, and this idea is applied to satellite network and data center network in (Du et al., 2016) and (Zannettou et al., 2016) respectively. In (Hussein et al., 2017), a new routing scheme is proposed to achieve optimal bandwidth utilization for a MPTCP connection, where modules in SDN compute routings and keep track of available bandwidth. However, all above solutions only take the currently processed connection into consideration and completely ignores the other connections in network, so they can only mitigate collisions between subflows in the same connection and are powerless to collisions between different connections' subflows.

In (Jiang et al., 2018), a novel SDN-based MPTCP routing solution with load awareness is proposed for satellite network. It is capable of bottleneck link detection, and paths are selected for subflows according to the calculation result of available bandwidth. However, the computing formula for links' available bandwidth is unreasonable to some extent: The available bandwidth of a path is set to the maximum value of left bandwidth and average bandwidth. The average bandwidth only makes sense when the bandwidth of the bottleneck link is allocated equally for each subflow passing it, however, the load balancing of MPTCP aims to perform fair bandwidth allocation for each connection rather than for subflow. The calculation of available bandwidth ignores the bandwidth allocated for other subflows of competing connections, so it cannot reflect the fairness described in the resource pooling principle in essence.

A significant issue must be addressed before SDN-based MPTCP is used in large-scale data center networks. An expensive and power-hungry resource, Ternary Content Addressable Memory (TCAM), is required to store forwarding rules in switches, and more data flows generated by SDN-based MPTCP will increase the consumption of TCAM. To reduce the SDN-based MPTCP's demand for TCAM resource, Segment Routing (SR) technique are used in (Pang et al., 2017) and (Barakabitze et al., 2018) to decrease the number of forwarding rules in video service and data center respectively. Experiment results show that the flow table size diminishes remarkably when SR is enabled in networks. However, network overhead may be caused by SR and future research is essential to avoid this limitation.

Above-mentioned researches, which aim to the improve MPTCPs resource utilization with SDN, involve path management, route selection modules and forwarding flow table. However, other targets are neglected which should get the same attention including fairness and responsiveness in networks.

3. Design of S-MPTCP

As a holistic solution to improve MPTCP's efficiency, fairness and responsiveness comprehensively based on SDN, the control mechanism of S-MPTCP is composed of three parts: resource exploration, resource

Table 2
Connection information.

Connection ID	Subflow ID	Link	Expected Throughput
1	1	1, 2, 3	et_{11}
1	2	4, 5, 6	et_{12}
2	1	7, 5, 8	et_{21}
2	2	9, 10, 11	et_{22}
3	1	12, 10, 13	et_{31}
3	2	14, 15, 16	et_{32}

allocation and flow control. In this section, we present the design ideas of S-MPTCP and these three parts in detail.

3.1. Resource exploration

All the other work about resource pooling is on the foundation of resource exploration. In this step, an optimal subflows routes set for newly-established connection will be calculated and outputted to realize the fully resource exploration and maximize the resource utilization. We consider the following circumstance: In a SDN network, there exist several MPTCP connections and at some point, a server host plans to establish a new MPTCP connection to its remote client. The total utilized resource pool can be defined as the total bandwidth of existing connections and the new connection, and the main target of S-MPTCP in this stage is to maximize the utilized resource pool.

Based on above analysis, we divide the path selection of subflow into two stages: In the first stage, path selection will be performed to explore residual network resource which has excluded the resource shared by existing connections, and in the second stage, necessary extra paths will be allocated to the new connection's subflows for the reason of fairness.

Therefore, the task of the first stage is to select the optimal subflow routes for the new connection to maximize the bandwidth under the limitation of residual network capacity, which is the result of removing the expected occupied portion of existing connections.

In our mechanism, an important concept is the expected throughput of a subflow. It specifies how much bandwidth the subflow should occupy and this value is calculated in Section 3.2 according to the resource pooling principle. Suppose there is a set of connections C which comprises a set of subflows S in the network. et_{ij} is the expected throughput of a subflow S_{ij} which is the jth subflow of connection C_i. Each subflow S_{ij} is associated to a path P_{ij} which consists of a set of links L. $L_k \in P_{ij}$ denotes that this subflow passes through link L_k and b_k is the bandwidth of link L_k. An information table can summarize these connections and path information, as shown in Table 2.

With the help of this table, we can get a residual graph which contains the information about available paths and residual capacities of links. The residual bandwidth rb_k of link L_k is calculated to be the portion left over after subtract the value of expected throughputs of subflows which pass through link L_k from the original bandwidth b_k, as shown in the following formula:

$$rb_k = b_k - \sum_{L_k \in P_{ij}} et_{ij} \tag{1}$$

The rest task can be regarded as a maximum flow problem, which can be written into an optimization model:

$$\max\left(\sum_{(v,t)\in A} f_{vt}\right) \tag{2}$$

$$s.t. \quad 0 \le f_{uv} \le rb_{uv}$$

A is the arc set and $u, v, s, t \in A$ where s, t are the source and the sink respectively. This problem has been researched for several decades, and several efficient maximum flow algorithms have been proposed, including Ford-Fulkerson algorithm (Ford and Fulkerson, 1956), Edmonds-Karp algorithm (Edmonds and Karp, 1972), Dinic's algorithm (Dinic, 1970), dynamic tree blocking flow algorithm (Sleator and Tarjan, 1983, 1985), push-relabel method (Goldberg and Tarjan, 1988) (Some specific algorithms for the problem with unit capacities are not within the scope of this paper). Running time is the most significant indicator of maximum flow algorithms, and that of above algorithms are listed in Table 3, where n and m are the number of vertices and arcs in the residual network, and U denotes the largest link capacity. A crucial defect of Ford-Fulkerson algorithm and Edmonds-Karp algorithm is that they may not converge when capacities are not integral, so they are not ideal options in actual networks. According to (Goldberg and Tarjan, 2014), although dynamic trees can achieve the best worst-case bounds, they are not adopted in practical usage because most practical instances are relatively easy. Besides, the constant factors in dynamic tree implementations are relatively large. Finally, we choose Dinics algorithm to address maximum flow problem in this paper. According to the execution results of Dinic's algorithm, optimal subflow routing and their available bandwidth can be obtained with further processing.

Table 3
Maximum flow algorithms and running time.

Algorithm	Running Time
Ford-Fulkerson algorithm	$O(mU)$
Edmonds-Karp algorithm	$O(nm^2)$
Dinic's algorithm	$O(n^2m)$
Dynamic tree blocking flow algorithm	$O(nmlogn)$
Push-relabel method	On^2m

Table 4
Dinic's algorithm's running time in a K-ray FatTree topology.

K	Vertices	Arcs	Running Time/ms
4	36	48	0.0886
8	208	384	0.3721
16	1344	3072	8.2371
32	9472	24,576	427.8041

So is the running time of Dinics algorithm acceptable in networks? We measure the running time of Dinics algorithm with our C++ implementation. Measurement is performed in FatTree topology, which is probably the most popular structured datacenter topology, and bandwidths of links are set to be a random value between 0 Mbps and 10 Mbps. Table 4 shows the number of vertices and arcs and the average running time of Dinics algorithm in a K-ray FatTree topology. According to measurement results in Table 4, when there are hundreds of vertices and arcs in the network, the running time of Dinics algorithm is at a negligible level. When the number swells into thousands, the running time can be tolerated in most cases. So its reasonable to argue that for a medium-sized network, it wont take much of time for S-MPTCP to perform Dinics algorithm and complete initialization.

After the operations of the first stage, several paths may have been allocated for subflows of the newly-established connection. However, apart from resource exploration, fairness between new connection and existing connections also must be taken into account. Because in the first stage, resource occupied by existing connections is out of the range of new connection's resource exploration, the total bandwidth of allocated paths for new connection might be too low or even zero when existing connections use up the bandwidth of bottleneck links on available paths. In this case, the bandwidth compensation for the new connection in the following resource allocation procedure introduced in

Section 3.2 will be very limited and fairness between connections cannot be ensured. Therefore, we propose a solution to find extra paths for the new connection in the second stage. For each available path P_r, we will calculate the difference between the total amount of expected throughput above average level of connections which pass through links belonging to P_r and the average expected throughput:

$$oset_r = \sum_{\{i|P_i \bigcap P_r \neq \emptyset\}} \sum_j et_{ij} - \frac{numc_r}{numc} \sum_i \sum_j et_{ij} \tag{3}$$

$numc_r$ is the number of connections which pass through links belonging to P_r. A larger $oset_r$ means that more bandwidth resource can be vacated from existing connections related to this path and allocated to newly-established connection. Therefore, we calculate the $oset_r$ parameter for each available path in the original graph, and sort all paths based on it. Besides, sufficient path capacity is essential to decrease the throughput gap in the resource allocation stage. When the total bandwidth of selected paths reaches $\sum_i \sum_j et_{ij}/numc$, it will be enough bandwidth to achieve fair bandwidth allocation in the following step, and path selection ends at this point. The complete resource pool finding algorithm is shown in Algorithm 1.

Algorithm 1 Resource pooling exploration algorithm.

Input: b_k, rb_k
Output: subflows routes
Init: $PS = \emptyset$
Use Dinic's algorithm and a subflow path set PS is obtained
While $\sum_{P_r \in PS} \min_{L_k \in P_r}(b_k) < \sum_i \sum_j et_{ij}/numc$ **do**
  $maxoset = -\infty$, $maxr = 0$
  for each available subflow path P_n **do**
    $oset_n = \sum_{\{i|P_i \bigcap P_n \neq \emptyset\}} \sum_j et_{ij} - \frac{numc_n}{numc} \sum_i \sum_j et_{ij}$
    if $oset_n > maxoset$ **then**
      $maxoset = oset_n$, $maxr = n$
    end if
  end for
  $PS = \{PS, P_{maxn}\}$
end while
Set routes for subflows according to PS

3.2. Resource allocation

After the above operations, subflow paths have been identified which means the utilized resource pool shared by all connections has also been fixed. The next question is which manner should we adopt to allocate resource to these connections to ensure efficiency and fairness between connections?

$$\max \left(\sum_i \log \left(\sum_j et_{ij} \right) \right) \tag{4}$$
$$s.t. \quad A \cdot et \leq b, et \geq 0$$

The straightforward approach is to set and solve an optimization problem to balance efficiency and fairness. In (Kelly et al., 1998) and (Han et al., 2006), the above optimization problem drawn from Nash arbitration scheme was proposed, and it has been proved that throughputs of connections are desirable optimal, fair operating points for the individual connections, when they solve the optimization problem. However, a critical problem is that this optimization problem belongs to mixed integer non-linear programming which has been shown to be NP hard (Bonami et al., 2012). Thus, we propose a heuristic algorithm to perform resource allocation leading to a tradeoff efficiency and fairness in a short running time.

If two connections don't pass through a same link, these two connections are independent to each other in our concept. If there is a connection set composed of several connections, and a connection in this set is not independent to at least one connection in this set, and meanwhile it's independent to other connections which are not in this set, the resource of these collection set can be regarded to be independent to other connections. When new connection's subflow routings have been identified through the first stage processing, the connection set it belongs to is also been fixed, and expected throughputs of all connections in this set are our algorithm's processing objects to achieve efficient and fair resource allocation.

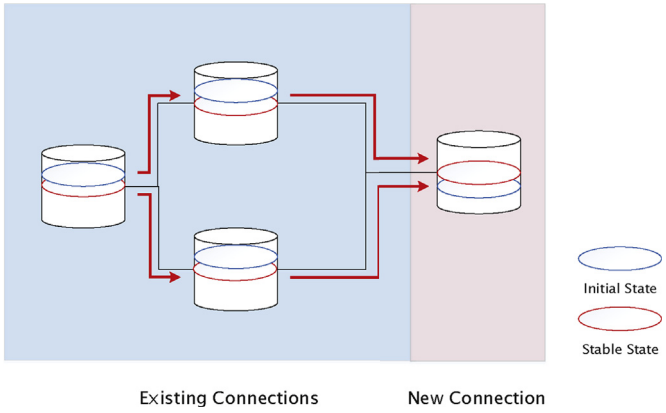


Fig. 5. The sketch of resource reallocation with water flow behavior.

A major goal of resource allocation is to achieve fairness, which means the bandwidth of connections in the same set should be as close as possible. This process can be naturally associated with the behaviors of water flows shown in Fig. 5: Connections can be seen as container and when two connections share common links, the two containers are interconnected. The water height of containers represents the expected throughput and the movement of water will drive the resource allocation. When water heights all converge to a stable state, we can consider that the resource has been allocated to each connection in a fair manner.

When a new connection is established in the SDN network, it can be regarded as a new container. If new connection's expected throughput is lower than the "average level", water flow will move from other connections in the same set until the new connection's expected throughput reaches the maximum value of average expected throughput and the upper limit of path bandwidth. We decompose this process into a series of independent operations on single connection, rather than perform adjustments to all related connections simultaneously. The algorithm details are as follows.

The preparatory work falls into two parts: The first task is to construct two graphs, the residual graph for judgement and the residual graph for calculation. The judgement residual graph records the results that the expected throughputs of existing connections have been subtracted, while the residual graph for calculation is the result that only processed connections expected throughputs in the current process have been subtracted. When expected throughput of a connection changes, both graphs will be updated correspondingly.

Through the information stored in judgement residual graph and path routing of connections, connections which belong to the same set with the new connection can be found out and marked. The relationship between connections in the set falls into two categories: direct joint and indirect joint, while relationship between subflows and connections in the set falls into three categories: direct joint, indirect joint and disjoint. All these relationships are recorded and up to this point, the preparation work is finished.

Let's first go into the process of the newly-established connection. A parameter we are interested in is the maximum throughput that the new connection can approach under the bandwidth limitation of judge-

ment residual graph. In the previous stage, all connections' routings are identified. According to judgement residual graph and the subflow path routing, the maximum achieved throughput of the connection $maxt_n$ and throughputs of subflows $maxt_{nj}$ at this point can be estimated using Algorithm 2, and we set these values as the initial values of expected throughput.

Algorithm 2 Maximum throughput estimation algorithm.

Input: link bandwidth b_k, residual bandwidth nb_k, number of subflows on link k $nums_k$
Output: $maxt_n$, $maxt_{nj}$
Init: $flag = 0$
for each subflow S_j **do**
  $initt_j = \min_{L_k \in P_j} b_k$
end for
for each link L_k **do**
  $nb_k = \min\left(rb_k - \sum_{L_k \in P_j} init\_t_j, 0\right)$
end for
while $flag = = 1$ **do**
  $flag = 0$
  $bo_j = \sum_k (nb_k / nums_k)$
  $minbo = 0$, $minj = 0$
  for each subflow S_j **do**
    if $bo_j < minbo$ **then**
      $minbo = bo_j$, $minj = j$
    end if
  end for
  $ot = t_{minj}$
  $t_{minj} = \max\left(t_{minj} + \min_{L_k \in P_{minj}} (nb_k), 0\right)$
  for $L_k \in P_minj$ **do**
    $nb_k += ot - t_{minj}$
    if $nb_k < 0$ **then**
      $flag = 1$
    end if
  end for
end while
$maxt_n = \sum_j t_j$, $maxt_{nj} = t_j$

Algorithm 3 The process order of connections.

Find the next connection to process:
Input: *proqueue*
Output: *next_c*
Init: $mind = +\infty$
for each connection C_i **do**
  if $proqueue[i] > 0$ **and** $proqueue[i] < mind$ **then**
    $mind = proqueue[i]$
    $next\_c = i$
  end if
end for

Update the order after connection C_n is processed:
Input: *jointd*, *proqueue*
Output: *proqueue*
Init: $mind = +\infty$
for each connection C_i **do**
  if $jointd_{ni} = = 1$ **and** $proqueue[i] = = -1$ **then**
    $proqueue[i] = proqueue[n] + 1$
  end if
end for
$proqueue[n] = 0$

The initial expected throughput can be regarded as the current water height in the new container, and it will be compared with the average throughput to decide whether to balance throughput of other connections and the new connection. The average throughput is:

$$avgt = \left(\sum_i \sum_j et_{ij} + maxt_n\right) / (numc + 1) \tag{5}$$

where $numc + 1$ is the current number of connections in this set. If $maxt_n < 0.9 \cdot avgt$, the allocation is considered to be unfair for the new connection. $maxt_n$ and $maxt_{nj}$ are recalculated according to the calculation residual graph, and the increase of expected throughput is:

$$\Delta_n = \min(avgt, maxt_n) - \sum_j et_{nj} \tag{6}$$

If this is an actual water movement problem, we have finished the process of the new container. However, because MPTCP connection consists of several subflows, the increase of throughput must be allocated to subflows. Based on the previous preparation, the number $numjoint_{nj}$ and total throughput $etjoint_{nj}$ of connections which are joint with a certain subflow can be obtained. $totalet_{nj} = etjointnj - avgt \cdot numjoint_{nj}$ represents the total throughput of connections which are joint with the subflow that is above average. The greater the value, the more resource should be extracted from the connection subset and the subflow can get more throughput compensation. On the other hand, the compensation is limited by the link bandwidth on the subflow. Therefore, Δ_n is prorated recurrently according to $totalet_{nj}$ under the limitation of link bandwidth.

After the first connection process, both residual graphs are updated and next, the expected throughputs of connections which are directly joint to the newly-established connection need to be adjusted. Naturally, expected throughputs of connections which are joint with processed connections also need adjustment. These is a hierarchal relationship between connections in the set according to the distance to the new connection. We choose to adopt breadth first search to make amendments, and the process method for connections is same with the process of the new connection. When expected throughput adjustments end, the whole network enters steady state phase. The pseudo code is shown in Algorithm 3 and Algorithm 4.

3.3. Flow control

After above process complete, subflows have been established on the routing identified in the first step, and expected throughputs are set according to the above algorithms. In the final step, flow control on the side of host will be improved leveraging these expected throughput values.

A significant control method can be used in the Slow-Start phase. Our previous work has shown that the present Slow Start mechanism of MPTCP may induce serious congestion window (cwnd) overshoot leading to severe performance degradation. Due to lack of a reasonable ssthresh, the exponential growth of cwnd is kept until packet losses occur and when slow start stage ends, MPTCP subflows may seriously overshoot their cwnd beyond path capacity. Acute packet reordering problem caused by cwnd overshooting will likely result in throughput degradation and poor overall utilization, especially when there is noticeable path heterogeneity between MPTCP subflows.

With the help of SDN, a threshold can be set based on the expected throughput calculated before to control the Slow Start scheme to avoid the cwnd overshooting and the consequent bottleneck overflow. The basic idea is that when subflow's throughput approaches its expected value, this subflow will exit from Slow Start and switch to Congestion Avoidance. Because expected throughput has constrained the transmission under the limitation of link capacity, the phenomenon of cwnd overshooting will be eliminated. Additionally, efficiency and fairness of network can be well balanced when the subflow exits Slow Start.

After the subflow enters Congestion Avoidance, cwnd will keep linear increasing to explore the rest bandwidth. When traditional AIMD

Algorithm 4 Resource allocation algorithm.

Input: judgement residual graph, calculation residual graph
Output: et
Init: $proqueue[numc + 1] = 1$
While $next\_c$ **do**
  $n = next\_c$
  Update $jointc$, $jointd$, $etjoint$ and $numjoint$
  Calculate $maxt_n$ and $maxt_{nj}$ according to the judgement residual graph
  $avgt = \left(\sum_i \sum_j et_{ij} + maxt_n\right) / (numc + 1)$
  if $maxt_n \geq 0.9 \cdot avgt$ **then**
    for each subflow S_{nj} **do**
      $et_{nj} = maxtnj$, $et_n = maxt_n$
      $proqueue[n] = 0$
    end for
  else
    Calculate $maxt_n$ and $maxt_{nj}$ according to the calculation residual graph
    $\Delta_n = \min(avgt, maxt_n) - \sum_j et_{nj}$, $\Delta_{tmp} = \Delta_n$, $it = 0$
    if $avgt \geq maxt_n$ **then**
      for each subflow S_{nj} **do**
        $et_{nj} = maxtnj$, $et_n = maxt_n$
      end for
    else
      While $\Delta_{tmp} > 0.1 \cdot \Delta_n$ **and** $it < threshold$
        Calculate $maxt_n$ and $maxt_{nj}$ according to the calculation residual graph
        $sumet = 0$
        for each subflow S_{nj} **do**
          $totalet_{nj} = etjointnj - avgt \cdot numjoint_{nj}$
          if $maxt_{nj} > 0$ **and** $totalet_{nj} > 0$ **then**
            $sumet += totalet_{nj}$
          end if
        end for
        for each subflow S_{nj} **do**
          if $maxt_{nj} > 0$ **and** $totalet_{nj} > 0$ **then**
            $\Delta_{nj} = \min\left(\Delta_{tmp} \cdot \frac{totalet_{nj}}{sumet}, maxt_nj\right)$
            $et_{nj} += \Delta_{nj}$, $et_n += \Delta_{nj}$, $\Delta_{tmp} -= \Delta_{nj}$
          end if
        end for
        $it++$
      end while
    end if
  end if
  Update the judgement residual graph and the calculation residual graph
end While

congestion control algorithms of MPTCP (LIA, OLIA, BALIA, etc.) are adopted, the cwnd increase will last till subflow experiences a packet loss, and coupled cwnd reduction operation is performed. In other word, AIMD algorithms need to create losses to find the available bandwidth of the subflow which can cause performance penalty.

Now we have a chance to improve MPTCP congestion control module and address these mentioned problems. The ideal state of subflow's transmission rate in our design is to perform regular linear increase before it approaches expected throughput, and then maintain this level. When all subflows in network transfer data at the rate of expected throughput, the whole network will work in an efficient and fair manner. However, not all connections can always fully utilize their allocated bandwidth and in this case, part of resource will be wasted. Therefore, subflow's cwnd is permitted to increase after it approaches the expected throughput. At this point, the possibility of congestion increases, so the exploration of bandwidth should become more cautious and conservative. In the same time, the buffer queue length will be monitored periodically to anticipate congestion. The part of control mechanisms on hosts shown in Algorithm 5 are implemented in a plugin which adapts to current loss-based MPTCP algorithms.

A transmission rate evolution example is illustrated in Fig. 6. At time A, the subflow is established after handshake and cwnd experiences exponential growth in this stage. From time B to time C, the transmission rate approaches expected throughput, and subflow moves to Congestion Avoidance. The growth rate of cwnd is limited with parameter *rate_factor* and α. At time C, it is observed that a remarkable buffer queue is built and its length increases quickly, and parameter β serves

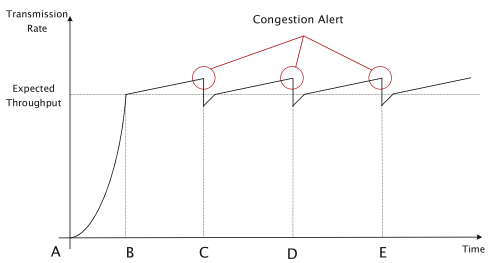


Fig. 6. The evolution of subflow's cwnd under S-MPTCP congestion control mechanism.

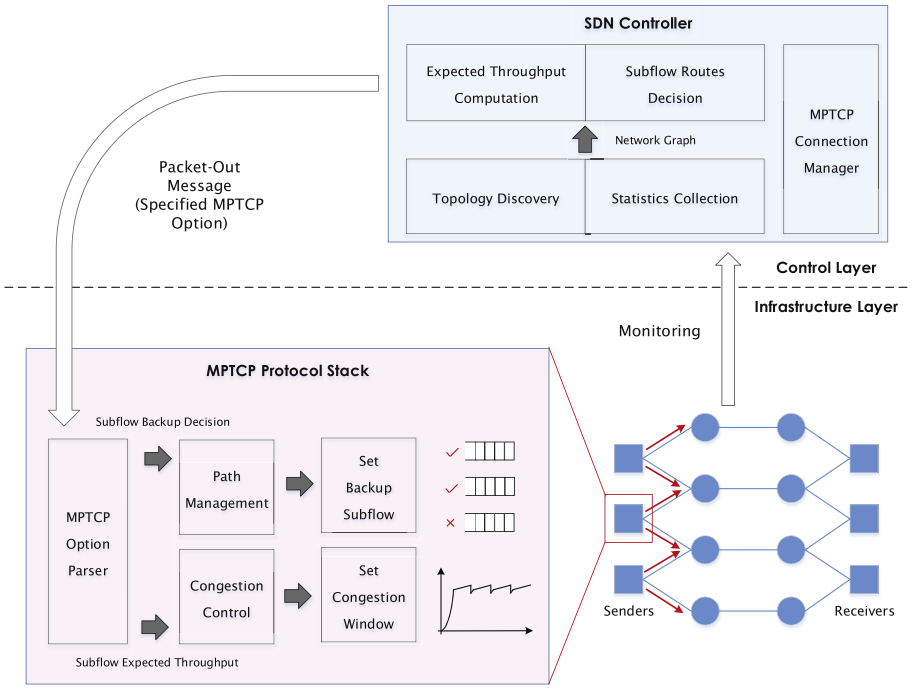


Fig. 7. The system architecture of S-MPTCP.

as a threshold. At this point, host will turn on the congestion alert that congestion is very likely to happen on the path. The alert will instruct the cwnd adjustment of host: The cwnd will decrease proactively if current throughput of subflow is higher than its expected throughput. The throughput after cwnd adjustment will be slightly lower than the level of expected throughput to drain buffer queue. Afterwards, the alert will be lifted and the increase of cwnd will return to regular linear increase. Through the above control mechanism, the transmission rate can stabilize around expected throughput, and prevention ability of congestion is ensured in the same time. The selection of parameters α and β will be discussed in Section 5.1.

Algorithm 5 Expected throughput based congestion control algorithm.

Init:$q_j = 0, rate\_factor = 1, \mu = 0.9$
On the end of round for subflow S_j
Check throughput t_j and current buffer queue length cq_j
if $t_j > et_j$ **then**
  $rate\_factor = \alpha$
  if $\beta \cdot cq_j \geq q_j$ **then**
    $cwnd_j = \mu \cdot et_j \cdot baseRTT_j$
  end if
else
  $rate\_factor = 1$
end if
$q_j = cq_j$
On each ACK of subflow S_j:
if $cwnd_j \leq ssthresh_j$ **and** $cwnd_j/rtt_j < et_j$ **then**
  perform regular slow start algorithm
else
  perform regular congestion avoidance algorithm, and get the increase of cwnd $cwnd\_inc$
  $cwnd_j += cwnd\_inc/rate\_factor$
end if

4. System architecture

The system overview of S-MPTCP is shown in Fig. 7. In SDN controller modules and MPTCP protocol stacks of hosts, several modules are designed to realize resource exploration and allocation for MPTCP connections.

4.1. SDN controller

As the basis of our operations, the information of network topology will be collected and updated by SDN controller. All these information will be consolidated into a network graph for later use of other modules. When establishment events of MPTCP connection and subflow happen in network, MPTCP connection manager will parse MPTCP options from corresponding handshake packets and update related information. If an MP_CAPABLE option is parsed, both a new connection entry and a new subflow entry will be added and information including IP addresses, ports and token which serves as a unique connection identification are recorded.

Next, resource exploration and allocation algorithms are performed according to network graph and existing connection information. As the output of algorithms, optimal subflow routes set and subflow's expected throughput will be calculated for each connection through algorithms introduced in Section 3.1 and 3.2. According to the subflow route sets, corresponding route control can be completed by controller indepen-

Table 5
System configuration in all of our experiments.

Configuration	Parameter
CPU type	Intel Core i5-6500
Number of CPU cores	4
Memory	16 GB
Storage	200 GB
Operating system	Ubuntu 14.04 LTS
Kernel version	Modified MPTCP v0.91.2
Mininet version	Mininet 2.2.2
SDN controller	Modified Floodlight v1.2

Table 6
Connection measurement result with different parameter α and β

α	β	Total Throughput/Mbps	SD/Mbps	Convergence Time/s
1	1	33.926	0.019	3.4
1	2	33.944	0.089	3.1
1	3	33.967	0.092	3.2
3	1	33.936	0.019	2.5
3	2	33.950	0.089	2.0
3	3	33.950	0.089	2.5
5	1	33.926	0.017	2.0
5	2	33.961	0.088	2.0
5	3	33.952	0.088	2.1
10	1	33.923	0.018	2.0
10	2	33.951	0.097	2.0
10	3	33.978	0.098	2.0

Table 7
Connection measurement result with different congestion control schemes.

Algorithm	Total Throughput/Mbps	SD/Mbps	Convergence Time/s
BALIA	33.962	3.511	13.8
OLIA	33.950	3.350	11.8
LIA	33.967	2.847	11.0
wVegas	33.123	0.992	6.2
S-MPTCP	33.950	0.018	2.0
Ideal Value	36	0	0

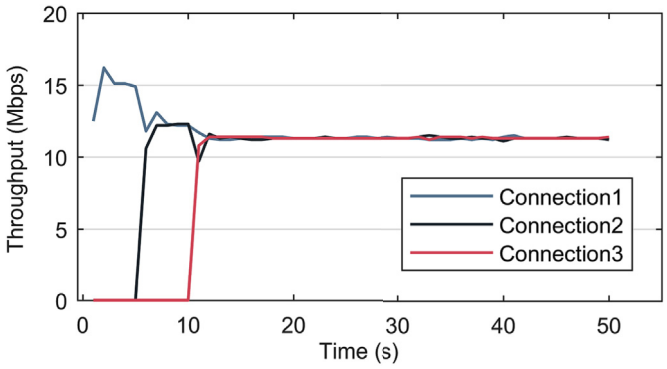


Fig. 8. Throughput trajectories of competing connections with S-MPTCP.

dently. If there is no route in the set for current subflow, this subflow will be regarded as backup subflow and its expected throughput will be set to zero.

Different from subflow route control, the congestion control is performed in MPTCP protocol stack of hosts. Therefore, controller needs to send the value of expected throughput to corresponding host after calculation. To inform the host of the subflow expected throughput, SDN controller will compose and send a TCP packet to host leveraging the Packet-out message of Openflow. The IP addresses and ports can be obtained from information stored in MPTCP connection manager. We design a new MPTCP option called MP_ET which carries the value of expected throughput. This packet will be sent out to the switch and finally received by host which the subflow belongs to.

4.2. MPTCP protocol stack

In Section 3.3, we propose a novel congestion control mechanism based on expected throughput. The main purpose of modules implemented in MPTCP protocol stack of hosts is to achieve congestion window control introduced in algorithms.

In essence, the Packet-out message is a single TCP packet, so it doesn't belong to any connection. Therefore, regular processing in TCP stack cannot apply to this packet. For this reason, before TCP stack demultiplexes arriving packets and binds them to connections, we add an additional MPTCP option parse operation, and after that, the packet is simply discarded if it contains MP_ET option. When the excepted throughput extracted from MP_ET option is zero, the corresponding subflow will be set to low priority.

Our mechanism is realized as a plugin which cooperates with existing congestion control algorithms in kernel. Two important criterions decide how to set cwnd to a proper level: The first one is the comparison between current throughput and expected throughput, and another is the growth rate of buffer queue length. Both parameters can be col-

Table 8
Connection measurement result with different delays.

Solution	Delay/ms	Total Throughput/Mbps	SD/Mbps	Convergence Time/s
S-MPTCP	15	33.950	0.018	2.0
	30	33.940	0.048	2.0
	100	33.950	0.030	3.0
	200	33.907	0.028	5.0
	500	33.795	0.039	13.0
regular MPTCP	15	33.962	3.511	13.8
	30	33.950	1.970	9.0
	100	33.933	3.406	13.0
	200	33.981	3.956	13.0
	500	34.131	3.271	24.0
Ideal Value		36	0	0

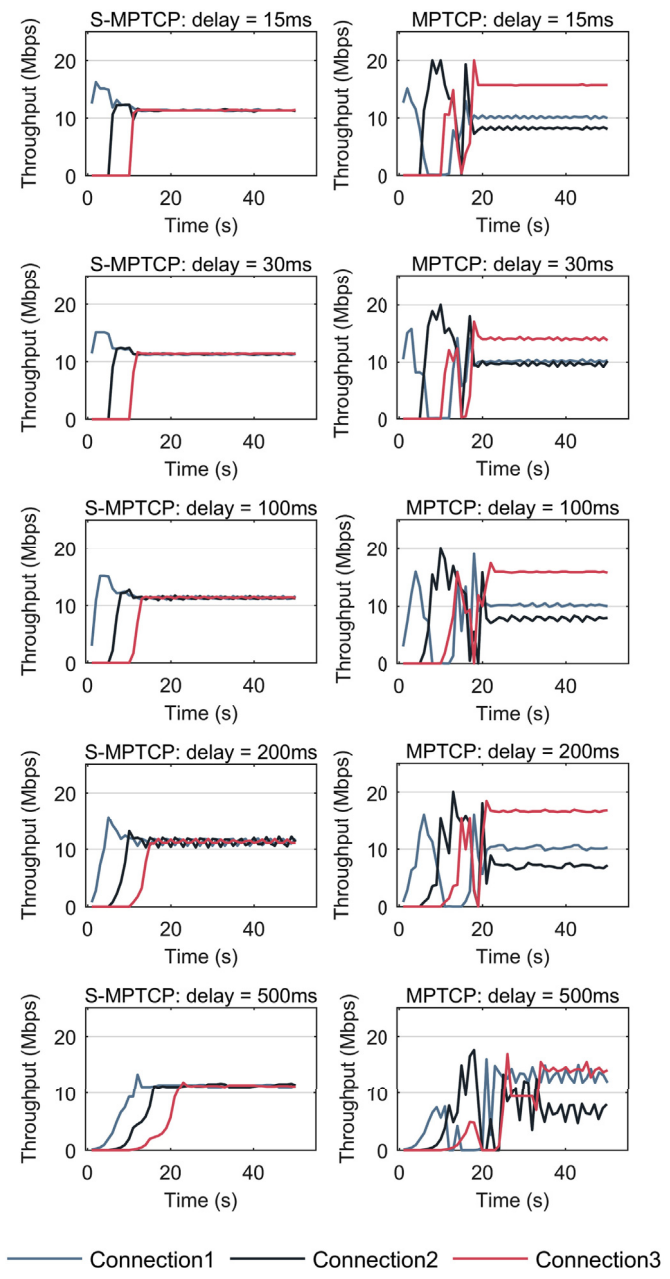


Fig. 9. Throughput trajectories of competing connections of S-MPTCP and regular MPTCP with different delays.

lected in controller or estimated in MPTCP stack. If we choose to collect throughput and buffer statistics by controller and then send them to hosts, it will bring significant cost of computational resource and link bandwidth resource to maintain statistics update in a timely manner. In the same time, the packet transmission delay will also hurt the effect of control schemes. For these reasons, the estimation of throughput and buffer queue length will be performed in MPTCP protocol stack. Based on these statistics and the information extracted from MP_ET option, expected throughput based slow start algorithm and congestion control algorithm can be implemented and aggregated with existing congestion control schemes.

5. Evaluation

In this section, we will evaluate the performance of S-MPTCP based on our system implementation. The SDN controller is implemented based on Floodlight v1.2. Modules including MPTCP connection manager, subflow routes decision and expected throughput computation are added and cooperate with existing modules. On the protocol stack on the hosts, MPTCP v0.91.2 is modified to realize functions introduced before. Mininet 2.2.2 is used to create out test topologies. The modification in the MPTCP congestion control scheme is based on BALIA algorithm in our experiment, which can also be performed on LIA and OLIA. The default congestion control algorithm of regular MPTCP is also set to BALIA for a direct comparison. All of our experiments are performed on the same testbed, and its configuration in all these experiments is shown in Table 5.

The evaluation is divided into two parts: First we will test the effect of load balancing in S-MPTCP and find the optimal parameter setting. In the second part, the improvement of resource utilization due to the routing solution in S-MPTCP will be examined. These two parts' results will be compared to the performance of regular MPTCP in Section 2.2.

5.1. Resource allocation test

First we test the load balancing effect of S-MPTCP and examine whether it achieves resource allocation with efficiency, fairness and responsiveness. In Section 2.2, regular MPTCP performs poorly on load balancing. The same test in Section 2.2 is performed for S-MPTCP with different settings of parameter α and β used in Algorithm 5 on the topology shown in Fig. 1, and results are shown in Table 6. When β increases from 1, the standard deviation rises dramatically, which means worse fairness achieved by load balancing with larger β. The convergence time is mainly affected by α, and results show that when α is greater than 5, it takes about 2 s for S-MPTCP to complete the whole load balancing. To sum up, $\alpha = 5$ and $\beta = 1$ are the optimal parameter setting according to our test.

Performance of S-MPTCP and regular MPTCP are presented together in Table 7 for a direct comparison. The total throughputs achieved by S-MPTCP and regular MPTCP are very close, while other indicators are improved tremendously with S-MPTCP. Standard deviation between connections' throughputs with S-MPTCP is reduced to 0.018, which approaches the ideal value. It means that resources are allocated to each connection almost equally. In terms of responsiveness, the convergence time due to load balancing is cut to about 2 s, which includes the time required for slow start stage. In fact, the load balancing effect after only 1 s reaches the effect with wVegas which costs 6.2 s. The high level of responsiveness ensures short and medium flows can enjoy control effect achieved by S-MPTCP. The throughput behavior of S-MPTCP is shown in Fig. 8, which gives a visual representation of load balancing with fairness and responsiveness.

A significant factor that influences MPTCP's performance is delay. The total delay of links on a subflow's path is varied from 15 ms to 500 ms. Table 8 and Fig. 9 show the performance of S-MPTCP and regular MPTCP under different transmission delay. With the increase of delay, the convergence time of S-MPTCP rises and it reaches 13s when delay is set to 500 ms, while the throughput and standard deviation are not obviously influenced by delay. Therefore, the rise of delay only degrades the responsiveness of S-MPTCP when efficiency and fairness are not affected. S-MPTCP outperforms regular MPTCP significantly according to experiment results, irrespective of delay.

5.2. Resource exploration test

In Section 2.2, we analyze the insufficient bandwidth utilization problem due to randomized routing in regular MPTCP. In this part, we investigate whether S-MPTCP can address the routing problem and fully utilize available resource. As a comparison, we also test the utilization of regular MPTCP and improved MPTCP in (Sandri et al., 2015)

Table 9
Connection throughput with the first kind of potential collisions.

Solution	Average Throughput/Mbps
Regular MPTCP	13.149
Disjoint MPTCP	18.816
S-MPTCP	18.862
Ideal Value	20

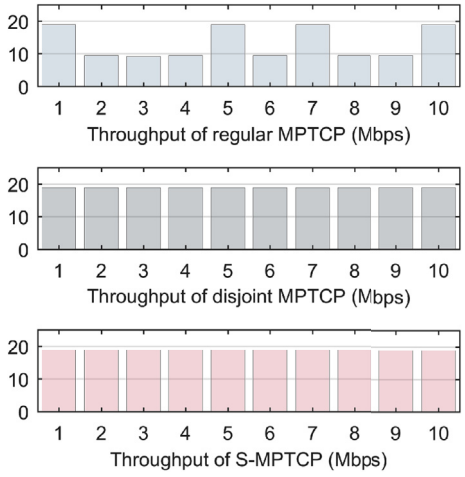


Fig. 10. Connection throughput with the first kind of potential collisions.

which chooses disjoint paths for subflows belonging to the same connection.

First we evaluate the resource utilization of these three solutions in the topology shown in Fig. 2(a) where collisions between subflows belonging to the same connection might happen. Measurements are repeated ten times and results are shown in Table 9 and Fig. 10. When connection adopts the routing solution of S-MPTCP and disjoint MPTCP, the bandwidth is fully utilized and collision is avoided. Therefore, both S-MPTCP and disjoint MPTCP can address this kind of collision problem effectively, while regular MPTCP will suffer throughput degradation frequently. When subflows are allocated to joint paths according to routing scheme of regular MPTCP, throughput is limited to half typical levels.

The second kind of potential collisions happen between subflows belonging to different MPTCP connections. Experiment is performed on the topology in Fig. 2(b) to study performance of these three solutions. From experiment results shown in Table 10 and Fig. 11, S-MPTCP can still utilize available bandwidth as much as possible. Unlike the first case, disjoint routing scheme fails to solve this kind of collisions because its routing algorithm only consider subflows' routings belonging to the same connection. Similarly, regular MPTCP connections cannot achieve stable and reasonable routing selection and consequently wastes vast amounts of network resources.

Table 10
Connection throughput with the second kind of potential collisions.

Solution	Average Throughput/Mbps
Regular MPTCP	30.926
Disjoint MPTCP	27.345
S-MPTCP	37.716
Ideal Value	40

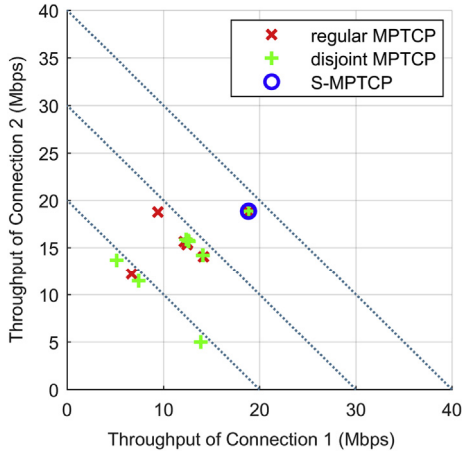


Fig. 11. Connection throughput with the second kind of potential collisions.

6. Conclusion and future work

To enhance the efficiency, fairness and responsiveness, we propose S-MPTCP which improves the resource exploration and resource allocation of MPTCP with the assist of SDN. Through modules we design in SDN controller and MPTCP stack, routing and load balancing problems of MPTCP are addressed effectively. Our experiments demonstrate that available bandwidth is fully utilized and allocated to each connection in a fair manner within a short time.

There are still some opening issues related to our systems and some new technologies:

Packet loss due to link error is an important issue in MPTCP, especially in lossy wireless networks. According to our observation, when packet loss rate is low (lower than 1%), S-MPTCP can maintain effective control and efficiency, fairness and responsiveness can be ensured. If loss rate rises to a high level, the control effect will be degraded. Therefore, if S-MPTCP is expected to perform better in lossy wireless network, specific control mechanism is needed for packet loss problem.

Nowadays, resources of downlink and uplink are in the nature of asymmetry for many emerging wireless communication technologies, such as Visible Light Communication (VLC) (Tanaka et al., 2001) and Millimeter Wave (mmWave). Powerful downlink and poor uplink are common characteristic in VLC and mmWave networks. In these cases, it's essential to consider the resource asymmetry in the calculation process of expected throughput to achieve finer-grained resource allocation.

In S-MPTCP, expected throughputs are adjusted according to our heuristic algorithm. machine learning, a compelling technology attracting attention in both scientific and industrial communities, is a potential solution for this adjustment task. Using captured transmission information from SDN controller and MPTCP stack, machine learning algorithms is likely to be trained to dig out hidden rules and get potential knowledge, and finally output an effective and particular solution. We expect machine learning can provide novel ideas for the cooperation of MPTCP and SDN.

In future work, we plan to investigate above issues to realize better cooperation between MPTCP and SDN, and achieve further performance enhancement.

Declarations of interest

None.

Acknowledgment

This work was supported by National Science and Technology Major Project of China MIIT (Grant No.2017ZX03001011-004), the Fundamental Research Funds for the Central Universities.

References

Abdelmoniem, A.M., Bensaou, B., Abu, A.J., 2017. Sicc: sdn-based incast congestion control for data centers. In: 2017 IEEE International Conference on Communications (ICC). IEEE, pp. 1–6.

Allman, M., Paxson, V., Blanton, E., 2009. Tcp Congestion Control. Tech. rep.. .

Bao, J., Wang, J., Qi, Q., Liao, J., 2018. Ectcp: an explicit centralized congestion avoidance for tcp in sdn-based data center. In: 2018 IEEE Symposium on Computers and Communications (ISCC). IEEE, pp. 00347–00353.

Barakabitze, A.A., Mkwawa, I.-H., Sun, L., Ifeachor, E., 2018. Qualitysdn: improving video quality using mptcp and segment routing in sdn/nfv. In: 2018 4th IEEE Conference on Network Softwarization and Workshops (NetSoft). IEEE, pp. 182–186.

Bonami, P., Kilin, M., Linderoth, J., 2012. Algorithms and software for convex mixed integer nonlinear programs. In: Mixed Integer Nonlinear Programming. Springer, pp. 1–39.

Brakmo, L.S., Peterson, L.L., 1995. Tcp vegas: end to end congestion avoidance on a global internet. IEEE J. Sel. Area. Commun. 13 (8), 1465–1480.

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

Chattopadhyay, S., Shailendra, S., Nandi, S., Chakraborty, S., 2018. Improving mptcp performance by enabling sub-flow selection over a sdn supported network. In: 2018 14th International Conference on Wireless and Mobile Computing, Networking and Communications (WiMob). IEEE, pp. 1–8.

Cordero, J.A., 2016. Multi-path tcp performance evaluation in dual-homed (wired/wireless) devices. J. Netw. Comput. Appl. 70, 131–139.

Dinic, E.A., 1970. Algorithm for solution of a problem of maximum flow in networks with power estimation. In: Soviet Math. Doklady, vol. 11, pp. 1277–1280.

Du, P., Nazari, S., Mena, J., Fan, R., Gerla, M., Gupta, R., 2016. Multipath tcp in sdn-enabled leo satellite networks. In: Military Communications Conference, MILCOM 2016-2016 IEEE. IEEE, pp. 354–359.

Edmonds, J., Karp, R.M., 1972. Theoretical improvements in algorithmic efficiency for network flow problems. J. Assoc. Comput. Mach. 19 (2), 248–264.

Ford, L.R., Fulkerson, D.R., 1956. Maximal flow through a network. Can. J. Math. 8 (3), 399–404.

Ford, A., Raiciu, C., Handley, M., Bonaventure, O., 2013. Tcp Extensions for Multipath Operation with Multiple Addresses. Tech. rep.. .

Goldberg, A.V., Tarjan, R.E., 1988. A new approach to the maximum-flow problem. J. Assoc. Comput. Mach. 35 (4), 921–940.

Goldberg, A.V., Tarjan, R.E., 2014. Efficient maximum flow algorithms. Commun. ACM 57 (8), 82–89.

Han, H., Shakkottai, S., Hollot, C.V., Srikant, R., Towsley, D., 2006. Multi-path tcp: a joint congestion control and routing scheme to exploit path diversity in the internet. IEEE/ACM Trans. Netw. 14 (6), 1260–1271.

Hopps, C., 2000. Analysis of an Equal-Cost Multi-Path Algorithm. Tech. rep.. .

Hussein, A., Elhajj, I.H., Chehab, A., Kayssi, A., 2017. Sdn for mptcp: an enhanced architecture for large data transfers in datacenters. In: 2017 IEEE International Conference on Communications (ICC). IEEE, pp. 1–7.

Jiang, Z., Wu, Q., Li, H., Wu, J., 2018. scmptcp: sdn cooperated multipath transfer for satellite network with load awareness. IEEE Access 6, 19823–19832.

Kelly, F.P., Maulloo, A.K., Tan, D.K., 1998. Rate control for communication networks: shadow prices, proportional fairness and stability. J. Oper. Res. Soc. 49 (3), 237–252.

Khalili, R., Gast, N., Popovic, M., Le Boudec, J.-Y., 2013. Mptcp is not pareto-optimal: performance issues and a possible solution. IEEE/ACM Trans. Netw. 21 (5), 1651–1665.

Kukreja, N., Maier, G., Alvizu, R., Pattavina, A., 2016. Sdn based automated testbed for evaluating multipath tcp. In: Communications Workshops (ICC), 2016 IEEE International Conference on. IEEE, pp. 718–723.

Lai, Y.-C., Ali, A., Hossain, M.S., Lin, Y.-D., 2019. Performance modeling and analysis of tcp and udp flows over software defined networks. J. Netw. Comput. Appl. 130, 76–88.

Nam, H., Calin, D., Schulzrinne, H., 2016. Towards dynamic mptcp path control using sdn. In: NetSoft Conference and Workshops (NetSoft), 2016 IEEE. IEEE, pp. 286–294.

Paasch, C., Detal, G., Duchene, F., Raiciu, C., Bonaventure, O., 2012. Exploring mobile/wifi handover with multipath tcp. In: Proceedings of the 2012 ACM SIGCOMM Workshop on Cellular Networks: Operations, Challenges, and Future Design. ACM, pp. 31–36.

Pang, J., Xu, G., Fu, X., 2017. Sdn-based data center networking with collaboration of multipath tcp and segment routing. IEEE Access 5, 9764–9773.

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

Raiciu, C., Barre, S., Pluntke, C., Greenhalgh, A., Wischik, D., Handley, M., 2011. Improving datacenter performance and robustness with multipath tcp. In: ACM SIGCOMM Computer Communication Review, vol. 41. ACM, pp. 266–277.

Raiciu, C., Handley, M., Wischik, D., 2011. Coupled Congestion Control for Multipath Transport Protocols. Tech. rep.. .

Ramaboli, A.L., Falowo, O.E., Chan, A.H., 2012. Bandwidth aggregation in heterogeneous wireless networks: a survey of current approaches and issues. J. Netw. Comput. Appl. 35 (6), 1674–1690.

Sandri, M., Silva, A., Rocha, L.A., Verdi, F.L., 2015. On the benefits of using multipath tcp and openflow in shared bottlenecks. In: 2015 IEEE 29th International Conference on Advanced Information Networking and Applications (AINA). IEEE, pp. 9–16.

Singh, K.V., Gupta, S., Verma, S., Pandey, M., 2019. Improving performance of tcp for wireless network using sdn. In: Proceedings of the 20th International Conference on Distributed Computing and Networking. ACM, pp. 267–276.

Sleator, D.D., Tarjan, R.E., 1983. A data structure for dynamic trees. J. Comput. Syst. Sci. 26 (3), 362–391.

Sleator, D.D., Tarjan, R.E., 1985. Self-adjusting binary search trees. J. Assoc. Comput. Mach. 32 (3), 652–686.

Tanaka, Y., Komine, T., Haruyama, S., Nakagawa, M., 2001. Indoor visible communication utilizing plural white leds as lighting. In: Personal, Indoor and Mobile Radio Communications, 2001 12th IEEE International Symposium on, vol. 2. IEEE (FF).

Wischik, D., Handley, M., Braun, M.B., 2008. The resource pooling principle. Comput. Commun. Rev. 38 (5), 47–52.

Wischik, D., Raiciu, C., Greenhalgh, A., Handley, M., 2011. Design, implementation and evaluation of congestion control for multipath tcp. In: NSDI, vol. 11, 88.

Yedugundla, K., Ferlin, S., Dreibholz, T., Alay, ., Kuhn, N., Hurtig, P., Brunstrom, A., 2016. Is multi-path transport suitable for latency sensitive traffic? Comput. Network. 105, 1–21.

Zannettou, S., Sirivianos, M., Papadopoulos, F., 2016. Exploiting path diversity in datacenters using mptcp-aware sdn. In: Computers and Communication (ISCC), 2016 IEEE Symposium on. IEEE, pp. 539–546.

Yanbing Liu received his B.S. degree from the Department of Electronic Engineering and Information Science (EEIS), University of Science and Technology of China (USTC), in 2017, and he is pursuing for his M.S. degree in USTC. His research interests include network protocols and multipath communication.

Xiaowei Qin received the B.S. and Ph.D. degrees from the Department of Electrical Engineering and Information Science, University of Science and Technology of China (USTC), Hefei, China, in 2000 and 2008, respectively. Since 2014, he has been a member of staff in Key Laboratory of Wireless-Optical Communications of Chinese Academy of Sciences at USTC. His research interests include optimization theory, service modeling in future heterogeneous networks, and big data in mobile communication networks.

Ting Zhu received the B.S. degree in electronic information from Huazhong University of Science and Technology, Wuhan, China, in 2015. He is currently pursuing the Ph.D. degree in electronic engineering and information science at University of Science and Technology of China. His current research interests include multipath communication, network protocols, and wireless network QoS.

Xiaohui Chen received the B.S. and M.S. degree in communication and information engineering from University of Science and Technology of China (USTC), Hefei, China, in 1998 and in 2004, respectively. He is currently an associate professor at the Department of Electronic Engineering and Information System, USTC. His current research interests include wireless network QoS, mobile computing, MAC protocol, and traffic model.

Guo Wei received his B.S. degree in EE from University of Science and Technology of China (USTC) in 1983, and M.S., Ph.D degrees in EE from Chinese academy of Science in 1986 and 1991 respectively. He is currently a full professor at the Department of Electronic Engineering and Information System, USTC. His current research interests are wireless and mobile communications, wireless multimedia communications and mmwave communication system.