Performance Analysis of MPTCP Under High Load Based on SDN Environment

Jiahui Hou[1,2], Dinghui Hou[1,2], Bixin Tao[2], Hui Qi[1,2(✉)], Xiaoqiang Di[1,2,3], Weiwu Ren[1,2], and Ligang Cong[1,2]

[1] School of Computer Science and Technology, Changchun University of Science and Technology, Changchun, China
[2] Jilin Key Laboratory of Network and Information Security, Changchun University of Science and Technology, Changchun, China
qihui@cust.edu.cn
[3] Information Center, Changchun University of Science and Technology, Changchun 130022, China

Abstract. Multipath Transmission Control Protocol (MPTCP) is in a rapid development process. MPTCP allows a network node to utilize multiple network interfaces and IP paths at the same time. It can take full advantage of network resources and provide reliable transmission, which brings advantages to users in terms of performance and reliability. In order to study the performance of MPTCP under high load, this paper uses Mininet to create an SDN environment and compare the performance differences between MPTCP and TCP under high load, and simulates a common Web application network architecture. The performance of MPTCP under this architecture is tested and analyzed. The test results show that MPTCP performs better than traditional TCP under high load. Besides, its performance can be optimized furtherly by adjusting related parameters.

Keywords: MPTCP · Load test · SDN

1 Introduction

With the development of information technology, the online examination system has become an important part of college examinations. However, with the number of students increasing, the response speed of servers becomes much slower, and the examination system is overwhelmed by high load. These problems can be solved with the emergence of MPTCP. MPTCP is a new type of multipath transmission protocol that bridges the gap between multipath networks and single path transmission. It can be seen as an extension of the conventional TCP protocol, providing the ability for both parties to transmit data over multiple paths simultaneously and allowing for the use of multiple paths to transmit a single stream of data between two terminal hosts. The original idea of designing MPTCP is to use plural links in end-to-end communication, so as to increase

W. Li and D. Tang (Eds.): MOBILWARE 2020, LNICST 331, pp. 57–68, 2020.
https://doi.org/10.1007/978-3-030-62205-3_5

the transmission bandwidth, make full use of network resources and improve fault redundancy. SDN separates the control plane and the forwarding plane to improve key features such as system flexibility and scalability, which is the development trend of the future network. SDN network simulation is widely used in networks. For example, Mininet gained popularity for allowing Linux applications to run on emulated networks. This paper designs an MPTCP load test framework under high load performance in SDN environment that is used to test performance of the server and client in the network. At the same time, this paper provides a network framework that uses Mininet to simulate a common Web application and considers multi-threaded and multi-cycle testing. It also analyzes the performance of multipath transmission under this framework. Besides, the results of the experiment are analyzed in depth and the parameters are tuned to make the framework take full advantage of the performance of MPTCP.

The main contributions of this paper:

(1) The performance difference between MPTCP and TCP under high load is compared, and the results are analyzed.

(2) A common Web application network architecture is simulated in the SDN environment, and the multi-path transmission performance under this architecture is analyzed.

The other parts of this paper are structured as follows: The second section introduces the work of this paper and MPTCP. The third section introduces related work. The fourth section describes the test framework and environment for this paper. The fifth section analyzes the experimental results. The sixth section summarizes the paper and introduces future research directions.

2 Background

Standard TCP communication uses a pair of IP addresses to transfer ordered data packets between two communicating parties. Therefore, there is a limit to using one interface, even though multiple interfaces are available at each end of the TCP connection. While MPTCP can use multiple interfaces to establish sub-connection paths between peers at the same time. By making use of all available interfaces at both ends of the communication, MPTCP aims to increase the network's resilience to interface failures (unavailability) and increase throughput. The MPTCP path is defined by two parts, the client IP and the server IP address. The subflow in MPTCP is a single-path standard TCP in which the component obtains fragments from a packet scheduler sent to the receiver. MPTCP is implemented at the transport layer and is supported by the same sockets used for application layer transparency in standard TCP. Each subflow uses standard TCP headers, which is transparent to the network layer.

Table 1 is the traditional TCP protocol and the multipath MPTCP protocol. It can be seen that MPTCP is built on the TCP subflow.

The MPTCP layer is located below the application layer and above the TCP layer in the protocol stack. It provides a standard TCP interface for the application layer to hide multipath, and it needs to manage multiple TCP subflows. The

Table 1. TCP protocol and MPTCP protocol comparison.

Application layer	Application layer		
TCP	MPTCP		
	Subflow 1 (TCP)	Subflow 1 (TCP)	Subflow N (TCP)
Network layer	Network layer		

MPTCP layer has the functions of path management, data packet scheduling, subflow interface, and congestion control.

Load test refers to different network environments, such as LAN, WAN, ATM or their combination. Test traffic is generated by one or more network interfaces of the test equipment. Through the device or network under test, certain indicators of the network or device Tests are performed, mainly including throughput, latency, and frame loss. The premise of all this is the generation of data packets and data streams. For traditional networks, the transport layer load test is mostly targeted at TCP and UDP.

The general method of load testing is to gradually increase the system load, examine the performance change of the system in an overload environment, and find the maximum load that the system can bear without affecting other performance indicators of the system. The load test is performed before the user understands the system performance, response time, and processing power, thereby achieving the purpose of forecasting in advance. It can identify problems through system performance characteristics or behaviors to help improve system performance.

This paper is aimed at the multipath transmission protocol MPTCP. JMeter is used to load test the server under the high-load MPTCP environment.

3 Related Work

With the widespread use of Web applications, Web performance issues have begun to plague users. For example, when a Web page is opened, it often fails to display normally for a long time or even disconnects. Due to people's increasing demand, the network bandwidth cannot meet the application requirements. The way to solve the problem is that designers increase the link bandwidth. In addition to it increasing the link bandwidth by aggregating the bandwidth of multiple physical links is also worked. If the network bandwidth resources of different links can be integrated and used, the utilization of network resources will be increased, and this will also bring higher throughput and better redundancy to the network.

Many researchers have proposed various methods, such as Reliable Multiplexing Transport Protocol, Parallel TCP [1,2] MPTCP [3] (Multipath TCP), and CMT-SCTP [4,5] (Concurrent Multipath Transfer-Stream Control Transmission Protocol). The current MPTCP and CMT-SCTP multipath transmission methods are relatively mature and perfect, but before large-scale deployment and

implementation of these methods, a large number of tests still need to be performed to observe whether they can obtain better performance in the existing network structure framework. MPTCP performance testing is very vital for the IETF standardization process. In this paper, the performance of MPTCP under high load is tested and analyzed.

MPTCP has the functions of path management, packet scheduling, sub-flow interface and congestion control. Researchers perform MPTCP performance tests from these aspects. Jinsung Lee et al. [6] considered the MPTCP neutron flow. The slow start (SS) and congestion avoidance (CA) stages accurately perform MPTCP performance analysis, and simulate a real MPTCP network in NS-3. The test results show that the method can accurately predict MPTCP throughput. Vivek Adarsh et al. [7] performed performance tests on MPTCP on heterogeneous subflow paths, focusing on the performance of MPTCP on paths with different characteristics, especially delay and loss rates. Experimental results show that it is difficult to implement MPTCP by using heterogeneous sub-paths. All sub-paths of MPTCP need to be close to the same network conditions in order to make full use of its potential. Yeonsup Lim et al. [8] The experimental results show that the use of the default path scheduler will reduce performance when there are path differences, and the paper proposes a new path scheduling method. The result is better than the existing scheduler. Xavier Corbillon et al. [9] measured the multiple transmissions of 100 MB files from the server to client by deploying MPTCP client and Web server, recording the time of data packet transmission path, and transmission on the network the time of the data packet and the time of the client receiving the data packet were analyzed on the MPTCP network. Feng Zhou et al. [10] deployed the MPTCP protocol to Linux and calculated the send and receive buffer sizes required by MPTCP in the actual Internet test platform NORNET, and tested the network by adjusting the buffer size. The experimental results show that MPTCP can increase payload throughput. Lucas Chaufournier et al. [11] deployed MPTCP in a Linux server and performed performance tests in a data center network. The performance of MPTCP under different congestion control algorithms was tested and compared with TCP. The comparison shows that MPTCP is feasible in the data center network. Slawomir Przylucki et al. [12] used multipath to study a multipath video real-time monitoring system based on an adaptive streaming mechanism. Its transmission network consists of WIFI and LTE units, and it is applied to the operation of a video surveillance system. Jiawei Huang et al. [13] used NS2 simulation test analysis to adaptively adjust the number of sub-flows through real-time network state and transfer the traffic from the congested path to reduce the delay. Yannis Thomas et al. [14] proposed a normalized multipath congestion control algorithm to achieve TCP-friendliness, and experimental evaluation using htsim simulator and real Linux proved that the method can speed up throughput.

At the same time, some researchers have applied MPTCP to the actual environment, studied the performance of MPTCP, and improved the performance of MPTCP from different angles. Karl-Johan Grinnemo et al. [15] used MPTCP to

reduce latency and improve the quality of service based on cloud applications. By researching three applications, namely Netflix, Google Maps and Google Docs, they represent typical applications that generate high-intensity, medium-intensity and low-intensity traffic. The results show that MPTCP can significantly reduce the latency of cloud applications, especially for applications such as Netflix and Google Maps. Tongguang Zhang et al. [16] applied MPTCP to an ad hoc network composed of drones or mobile devices. Run MPTCP on drones or MSDs. By improving the MPTCP sub-path establishment algorithm, it has a good performance in improving data throughput. Yuqing Qiu et al. [17] applied MPTCP to the migration of LXC containers. Because the container migration process needs to go through a WAN, congestion or network failure may occur. Therefore, MPTCP protocol is used to solve this problem and improve the flexibility of the migration process, reducing migration time.

Combining MPTCP with SDN is also a good method, which can take advantage of network resources and reduce the occurrence of network congestion. Pengyuan Du et al. [18] studied the performance of MPTCP on LEO satellite networks supporting SDN, and designed an SDN controller that identifies MPTCP substreams attached to the same MPTCP session and splits them into Intersecting paths. The SDN architecture centralizes routing logic, so the system is more scalable and minimizes onboard processing. Simulation results verify the effectiveness of the framework. Behnaz Arzani et al. [19] discussed the effect of initial sub-path selection on performance. And empirical data is used to prove which sub-path to choose to start the MPTCP connection may produce unintuitive results. Subsequently, the numerical results and models confirm the empirical results, which helps to design a better MPTCP scheduler. Navin Kukreja et al. [20] designed an SDN-based MPTCP path manager, implemented an automated test platform for performance evaluation, and analyzed different scheduling algorithms in-depth, and learned how the delay affects the MPTCP protocol. Overall performance. In order to more easily establish the MPTCP simulation environment, Matthieu Coudron et al. [21] implemented MPTCP in the NS3 network simulator, and compared it with the implementation of MPTCP in Linux, confirming that their effects are the same, and that NS3 is used in traffic processing. Aspect has more advantages.

Based on the research on the preliminary work, this paper finds that under high load conditions, there is still a lack of research and analysis of MPTCP performance. In this paper, we conducted a load test and analysis of mptcp in the SDN environment for the performance of mptcp under high load.

4 Experimental Architecture Design

In order to test the performance of MPTCP in a high-load, we simulated a common Web application network framework in the SDN environment for testing.

Figure 1 shows the simulated network architecture in the SDN environment. The Web server is deployed on H2 and the database is deployed on H3. The H1 simulated client is used to access the server and the records are stored in the H3 database. Each host has two IP addresses.

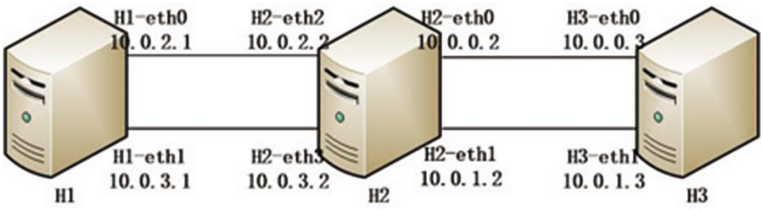


Fig. 1. A common Web application network architecture.

In this section, we will introduce the choice of platform implementation and testing schemes in detail.

Mininet [22]: It is a network emulator connected by some virtual terminal nodes, switches and routers, which can easily create a network supporting SDN. It uses lightweight virtualization technology to make the system comparable to real networks. Mininet uses process-based virtualization and network namespaces to create virtual networks, which are available in the latest Linux kernel. Programs running on Linux can basically run on Mininet, such as Wireshark.

JMeter [23]: The ApacheJMeter^{TM} application is open source software and is a Java application designed to load test functional behavior and measure performance. It was originally designed for testing Web applications, but later expanded to other testing capabilities. Apache JMeter can be used to test the performance of static and dynamic resources, Web dynamic applications. In this paper, it is used to simulate heavy loads on servers and network objects to test their strength or analyze the overall performance under different loads types.

The purpose of the experiment is to study the performance of MPTCP under the common Web application network architecture in the SDN environment. Through the comparison results of MPTCP and TCP in high load test, the performance of MPTCP is analyzed, and the server parameters are adjusted according to the results.

This paper compiled MPTCP on Ubuntu 18.04. The MPTCP scheduler is set as the default scheduler, which makes packet scheduling decisions based on the RTT of the subflow. The congestion control algorithm is CUBIC, which is the default Linux congestion control algorithm.

This paper uses Mininet to simulate the SDN network, the details are as follows:

(1) Under real network conditions, this paper record test cases by visiting the online examination system.

(2) Simulate the network architecture shown in Fig. 1 in the SDN environment. Three hosts H1, H2, H3, deploy servers and databases on H2, H3, and set the link bandwidth to 50M.

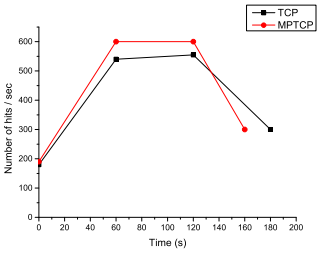
(3) Use JMeter to test and compare server access.

5 Analysis

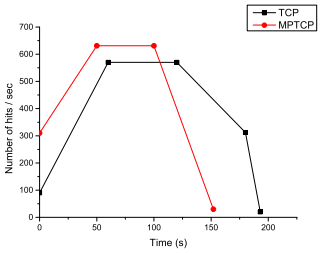
In this experiment, the high load test of MPTCP multi-path transmission and traditional single-path TCP was tested under the experimental architecture, and the multi-path MPTCP and single-path TCP transmission throughput were compared. Observed and analyzed the performance changes of MPTCP and traditional TCP under different loads.

In the throughput test, this paper fixed the number of requests, by reducing the thread startup time, the number of concurrent requests per unit time was increased to simulate the increase in load. In the test experiment, 4000 threads were fixed, and each thread initiated 20 requests. Figure 2 (a) to Fig. 2 (g) shows the actual throughput changes of single-path and multipath MPTCP with a thread start time of 160 s, 150 s, 140 s, 130 s, 120 s, 110 s and 100 s, respectively. Increasing experimental load by continuously reducing startup time. As can be seen from Fig. 2, the throughput of multipath MPTCP is significantly higher than that of traditional single-path TCP under each different loads. As the test load continues to increase, the throughput of multipath MPTCP also increases, and the throughput increases significantly. All MPTCP requests can be completed almost immediately after the thread startup time is over. From the minimum test load to the maximum test load, the throughput has increased by 50%, which has been greatly improved. The traditional single-path TCP transmission throughput also increases with increasing load, but the growth rate is not obvious, and the problem of erroneous requests appears, causing some requests to fail to respond normally. TCP takes longer than the startup time to complete all requests. From the minimum test load to the maximum test load, the throughput increased by only 16%. Figure 3 shows the throughput between hosts under different loads and protocols.

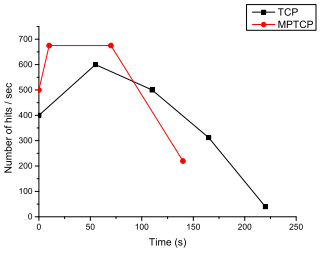
With increasing test load, traditional single-path TCP and multipath MPTCP show different results. Figure 4 shows the change in the number of error requests for traditional single-path TCP under increased load. (Multipath MPTCP increases with parallel requests, and no error requests occur, so the growth of single-path TCP's error requests is plotted separately in Fig. 4). The horizontal axis is the test load, and the vertical axis is the percentage of error requests. With the increase of test load, the throughput of traditional single-path TCP has increased slightly, but generally, it cannot meet the demand, resulting in the appearance of erroneous requests. As the number of wrong requests increases, the server needs extra time to process the wrong requests. As a result, the time for the server to respond to the wrong requests also increases, resulting in an increase in the total running time of the experiment. This is also the reason why the throughput increases but the total running time of the experiment also increases. Figure 5 shows the comparison of the total running time of traditional single-path TCP and multipath MPTCP experiments under different loads. The horizontal axis is the test load, and the vertical axis is the experimental running time. Multipath MPTCP can steadily increase throughput under increasing load without error requests. Because the server only needs to process normal requests,



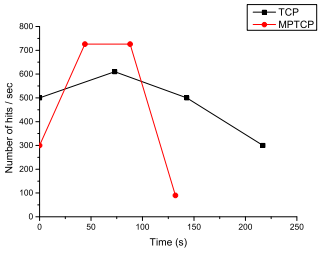
(a) Result 1



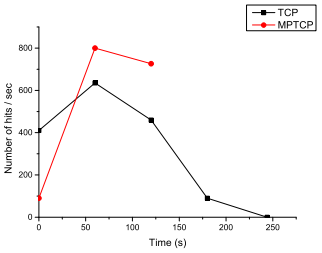
(b) Results 2



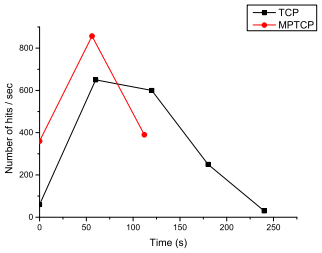
(c) Result 3



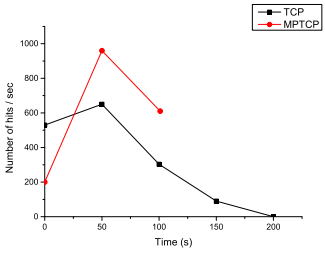
(d) Result 4



(e) Result 5

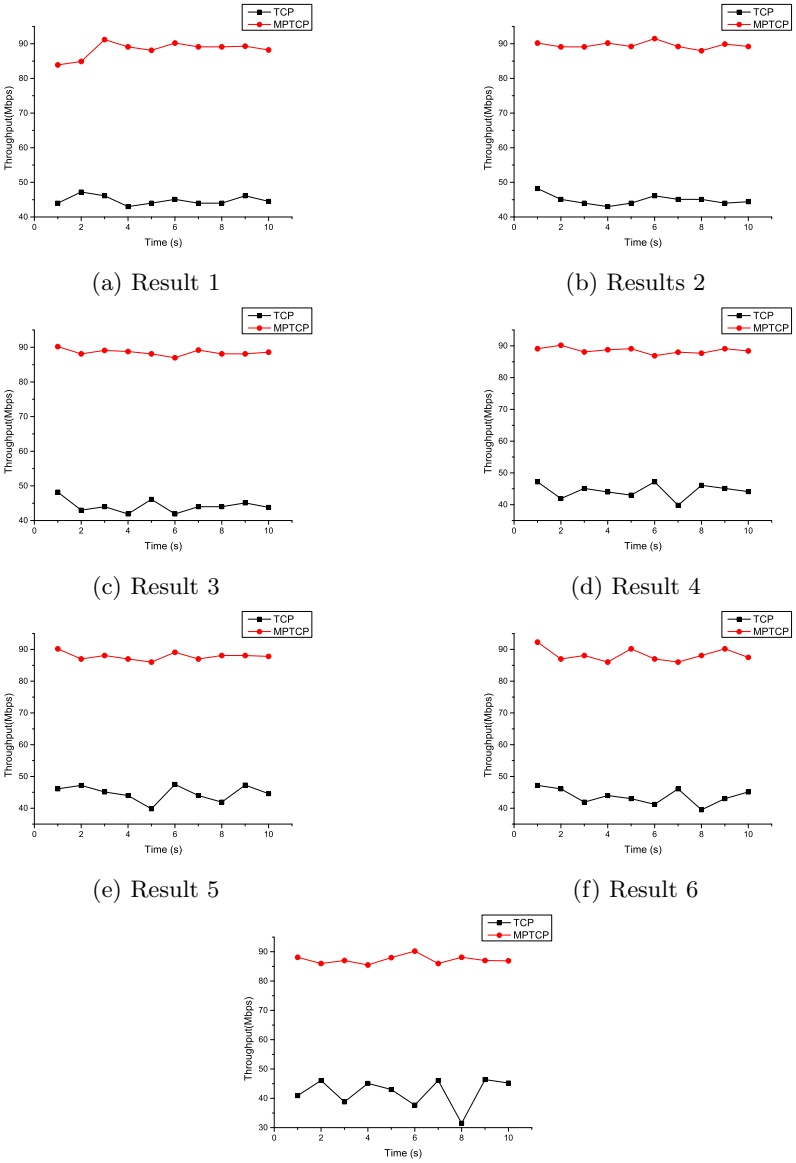


(f) Result 6



(g) Result 7

Fig. 2. Throughput comparison.



(a) Result 1

(b) Results 2

(c) Result 3

(d) Result 4

(e) Result 5

(f) Result 6

(g) Result 7

Fig. 3. Throughput between hosts under different loads.

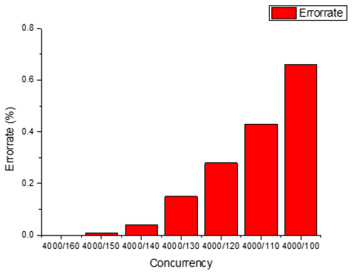


Fig. 4. TCP error rate.

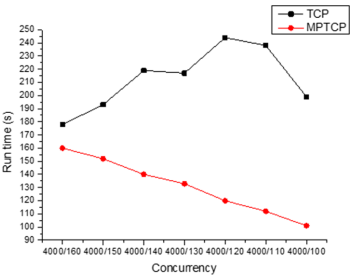


Fig. 5. Comparison of experiment running time.

the throughput increases with the increase of test load, and the experimental run time decreases with the increase of test load.

6 Conclusion and Future Work

This paper introduces MPTCP, uses the Linux kernel to compile MPTCP to implement MPTCP deployment, builds a common Web network framework in the SDN environment, tests the performance of MPTCP under high load and compares it with TCP. Through experimental results, we can see that under low or no load, the performance difference between multi-path MPTCP and traditional TCP is not much different. At the maximum load tested in this paper, the throughput of MPTCP is compared to traditional TCP has increased by 45%.

In future work, this paper considers applying MPTCP to data center networks, spatial information networks, and even combining with SDN, so that it can have better application scenarios, and is committed to improving the performance of MPTCP.

Acknowledgments. The manuscript was supported in part by the National Key Research and Development Program of China under Grant No. 2018YFB1800303 and the Science and Technology Planning Project of Jilin Province under Grant No. 20180414024GH.

References

1. Hacker, T.J., Noble, B.D., Athey, B.D.: Improving throughput and maintaining fairness using parallel TCP. In: IEEE INFOCOM 2004. IEEE, vol. 4, pp. 2480–2489 (2004)
2. Altman, E., Barman, D., Tuffin, B., et al.: Parallel TCP Sockets: Simple Model, Throughput and Validation. In: INFOCOM 2006, pp. 1–12 (2006)
3. Wikipedia. http://en.wikipedia.org/
4. Eklund, J., Grinnemo, K.J., Brunstrom, A.: Using multiple paths in SCTP to reduce latency for signaling traffic. Comput. Commun. **129**, 184–196 (2018)
5. Lai, W.K., Jhan, J.J., Li, J.W.: A cross-layer SCTP scheme with redundant detection for real-time transmissions in IEEE 80211 Wireless Networks. IEEE Access **7**, 114086–114101 (2019)
6. Lee, J., Im, Y., Lee, J.: Modeling MPTCP performance. IEEE Commun. Lett. **23**(4), 616–619 (2019)
7. Adarsh, V., Schmitt, P., Belding, E.: MPTCP Performance over Heterogenous Subpaths. In: 2019 28th International Conference on Computer Communication and Networks (ICCCN). IEEE, pp. 1–9 (2019)
8. Lim, Y., Nahum, E.M., Towsley, D., et al.: ECF: An MPTCP path scheduler to manage heterogeneous paths. In: Proceedings of the 13th International Conference on emerging Networking EXperiments and Technologies, pp. 147–159 (2017)
9. Corbillon, X., Aparicio-Pardo, R., Kuhn, N., et al.: Cross-layer scheduler for video streaming over MPTCP. In: Proceedings of the 7th International Conference on Multimedia Systems, pp. 1–12 (2016)
10. Zhou, F., Dreibholz, T., Zhou, X., et al.: The performance impact of buffer sizes for multi-path TCP in internet setups. In: 2017 IEEE 31st International Conference on Advanced Information Networking and Applications (AINA), pp. 9–16. IEEE (2017)
11. Chaufournier, L., Ali-Eldin, A., Sharma, P., et al.: Performance evaluation of Multi-Path TCP for data center and cloud workloads. In: Proceedings of the ACM/SPEC International Conference on Performance Engineering 2019, pp. 13–24 (2019)
12. Przylucki, S., Czerwinski, D., Sierszen, A.: QoE-oriented fairness control for DASH systems based on the hierarchical structure of SVC streams. In: Gaj, P., Kwiecień, A., Stera, P. (eds.) CN 2016. CCIS, vol. 608, pp. 180–191. Springer, Cham (2016). https://doi.org/10.1007/978-3-319-39207-3_16
13. Huang, J., Li, W., Li, Q., et al.: Tuning high flow concurrency for MPTCP in data center networks. J. Cloud Comput. **9**(1), 1–15 (2020)
14. Thomas, Y., Karaliopoulos, M., Xylomenos, G., et al.: Low latency friendliness for multipath TCP. IEEE/ACM Trans. Netw. **28**(1), 248–261 (2020)
15. Hurtig, P., Grinnemo, K.J., Brunstrom, A., et al.: Low-latency scheduling in MPTCP. IEEE/ACM Trans. Netw. **27**(1), 302–315 (2018)
16. Zhang, T., Zhao, S., Ren, B., et al.: Performance enhancement of multipath TCP in mobile Ad Hoc networks. In: 2017 IEEE 25th International Conference on Network Protocols (ICNP), pp. 1–2. IEEE (2017)
17. Qiu, Y., Lung, C.H., Ajila, S., et al.: LXC container migration in cloudlets under multipath TCP. In: 2017 IEEE 41st Annual Computer Software and Applications Conference (COMPSAC), vol. 2, pp. 31–36. IEEE (2017)
18. Du, P., Nazari, S., Mena, J., et al.: Multipath TCP in SDN-enabled LEO satellite networks. In: MILCOM 2016–2016 IEEE Military Communications Conference, pp. 354–359. IEEE (2016)

19. Arzani, B., Gurney, A., Cheng, S., et al.: Deconstructing MPTCP performance. In: 2014 IEEE 22nd International Conference on Network Protocols, pp. 269–274. IEEE (2014)
20. Kukreja, N., Maier, G., Alvizu, R., et al.: SDN based automated testbed for evaluating multipath TCP. In: 2016 IEEE International Conference on Communications Workshops (ICC), pp. 718–723. IEEE (2016)
21. Coudron, M., Secci, S.: An implementation of multipath TCP in ns3. Comput. Netw. **116**, 1–11 (2017)
22. Mininet. http://mininet.org/
23. JMeter. http://jmeter.apache.org/