Transparent Multipath: Using Double MPTCP Proxies to Enhance Transport Performance for Traditional TCP

Jiangping Han, Kaiping Xue, Wenjia Wei, Yitao Xing, Jianqing Liu, and Peilin Hong

Abstract

Multipath TCP (MPTCP) proxy is a way to provide multipath transmissions for current Internet hosts. However, previous works on MPTCP mainly focus on the scenario where one end of the connection is MPTCP-capable, which is not the case in current networks. In this article, we propose a transparent MPTCP (T-MPTCP) transmission scheme to achieve MPTCP transmission for TCP hosts at both ends. Specifically, T-MPTCP contains two components in the network: an aggregation box at the user side acting as an implicit proxy, and an aggregation server at the server side acting as an explicit proxy. The aggregation server can be deployed in different places of the network for optimized performance, and we thus propose a Proxy Location Selected (PLS) algorithm to determine the best aggregation server placement. To validate our design, T-MPTCP is deployed in a real network for experiments, whose results exhibit higher network throughput under T-MPTCP and further improvement with the PLS algorithm.

Introduction

Nowadays, network heterogeneity presents both challenges and opportunities to the current Internet. For instance, one communication device could be equipped with multiple network interfaces (e.g., a mobile phone with 3G/4G and WiFi interfaces), which makes it possible to use multiple paths for transmission at the same time. Clearly, multipath transmission not only aggregates multiple paths' capacity but also increases robustness and reliability of the connection. However, multipath transmission cannot be supported by the legacy TCP [1]. In light of this, some multipath transmission protocols have been proposed and MPTCP is one of the most promising protocols due to its good compatibility with the application layer [2].

As an extension to TCP, MPTCP has attracted much attention due to its support for concurrent use of multiple interfaces and compatibility with TCP. Technically, MPTCP splits data to concurrent transmissions through multiple interfaces, and each path between one pair of interfaces of two MPTCP hosts (client and server) is defined as a subflow. The MPTCP layer is added above TCP, which divides data of a connection into several portions and schedules them on parallel TCP subflows. In the MPTCP layer, many algorithms are proposed to enhance its performance. For example, path management algorithms determine how to initiate and control different subflows. Scheduling algorithms [3] distribute packets among different paths to keep in-order delivery. Furthermore, several congestion control algorithms have also been proposed to aggregate bandwidth, improve robustness and provide load balancing, while being friendly with traditional TCP [4]. The implementation of MPTCP only requires the modification of the transport layer while preserving the use of legacy TCP sockets on end hosts, which is transparent to the upper and lower layer stacks.

Although significant research efforts in MPTCP have been devoted to reduce the modification on host sides and provide transparency to upper-layer applications [5], the real deployment of MPTCP is still in its infancy. The most challenging part is to make all the end hosts support MPTCP, but due to a large amount of different devices, deploying MPTCP is a huge task. A natural question to ask is how we can modify the implementation of MPTCP so that the hosts can benefit from MPTCP without introducing huge burdens to them.

A promising idea to address this problem is to utilize a MPTCP proxy, which resides in the network and provides MPTCP support for MPTCP-incapable hosts. For example, a mobile phone equipped with 4G and WiFi interfaces and supports MPTCP could not gain benefit for a server which does not support MPTCP. But using a MPTCP proxy between them can help to make conversion of MPTCP and TCP, and provide bandwidth aggregation for MPTCP-incapable hosts. In [6], the first idea of MPTCP implicit proxy and explicit proxy is proposed. Thereafter, several other works focused on the design of different types of MPTCP proxies [7–11] and some of their performances are theoretically analyzed and experimentally tested in [12, 13].

However, these previous MPTCP proxy schemes only consider the situation where only one side of the connection is MPTCP-incapable but the other side should support MPTCP, as shown in Figs. 1a and 1b. They ignore the scenario where both end hosts are MPTCP-incapable, which is more common in current TCP/IP networks. Therefore, a novel solution is needed to achieve transparent multipath transmission in the general MPTCP-incapable network. In our previous work, we proposed TMPP [14] to provide mul-

Jiangping Han, Kaiping Xue (corresponding author), Wenjia Wei, Yitao Xing, and Peilin Hong are with the University of Science and Technology of China; Jianqing Liu is with the University of Alabama in Huntsville, Huntsville, AL 35899, USA.

Digital Object Identifier: 10.1109/MNET.112.1900648

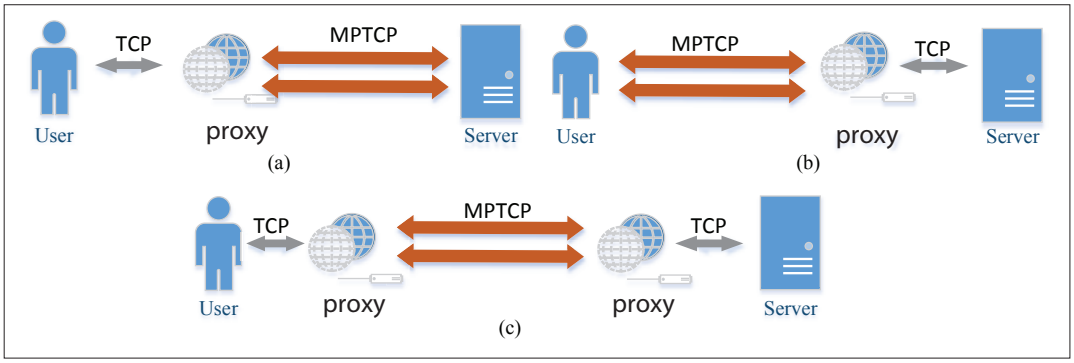

FIGURE 1. Different kinds of MPTCP proxies: a) user is MPTCP-incapable and the MPTCP proxy is used at the user side; b) server is MPTCP-incapable and the MPTCP proxy is used at the server side; c) both user and server are MPTCP-incapable and the MPTCP proxy is used at the both sides.

tipath transmission for two MPTCP-incapable hosts, which utilizes two implicit proxies. However, both implicit proxies should be deployed on the original path from the user to the server, which is not ideal for the flexibility of server-side proxies. Otherwise, TMPP needs to modify the original MPTCP protocol, which is too complicated to be realized.

In this article, we propose a simple but practical solution, Transparent MPTCP transmission (T-MPTCP), which targets the scenario of double MPTCP-incapable hosts, as shown in Fig. 1c. T-MPTCP adopts a double proxies scheme and takes advantage of both the implicit proxy and explicit proxy into account. It is compatible with the existing network architecture and all the MPTCP-incapable hosts do not need any changes. As shown in Fig. 1, T-MPTCP contains two devices in the network: the aggregation box and the aggregation server. The aggregation box is used at the user side, which can replace the user's traditional home router. It uses the modified Linux MPTCP kernel and acts as an implicit MPTCP proxy to convert TCP flow to MPTCP flow. Aggregation servers are used at the server side, which can be deployed in different rented cloud platforms. The deployed aggregation server acts as an explicit MPTCP proxy, which converts MPTCP to TCP for the content server through a universal explicit proxy protocol SOCKS and provides rate-up services.

The explicit proxy mode is more flexible for choosing a suitable aggregation server, which cannot only be deployed on the original transmission path, but also on other locations in the network. When the original path suffers from heavy traffic load or limited bandwidth, choosing the aggregation server with a suitable location will directly improve the transmission performance. For this reason, we further provide a Proxy Location Selection (PLS) algorithm as a supplement to our proposed T-MPTCP architecture, which is used to decide which aggregation server should be used to provide the best transmission gain based on the available path conditions.

Our contributions in this article can be summarized as follows. We introduce and implement a double MPTCP proxies solution called T-MPTCP, which uses an aggregation box at the user side and an aggregation server at the server side, to provide multipath transmission between them and improve the performance for users. This solution requires no changes on the original user and server. T-MPTCP includes a set of distributed aggregation servers to provide multipath transmission service for users. To further improve its performance, we propose a PLS algorithm which can optimize to select a suitable aggregation server for satisfactory bandwidth aggregation according to measured path conditions of each aggregation server. We deploy T-MPTCP in a real network to conduct the design verification and performance evaluation. The aggregation servers are deployed in different cloud servers provided by multiple cloud service providers. Experimental results show that T-MPTCP can efficiently aggregate the bandwidth of different paths between an aggregation box and an aggregation server, and achieve performance gains such up to 40 percent and 180 percent increased throughput compared with only using 4G telecom and 4G mobile, respectively.

The rest of the article is organized as follows. In the following section, we introduce some previous works of the MPTCP proxy. Then we present the overall architecture of our proposed system and the operation procedures. The details of the MPTCP proxy selection design are given following that. We then evaluate the proposed scheme via experiments. Finally, we present a discussion and draw a conclusion.

Overview of the MPTCP Proxy

In the original definition, MPTCP needs to be supported on both end hosts to achieve all its benefits. However, most of the current hosts do not support MPTCP. In light of this, several researchers have proposed and implemented different types of MPTCP proxies to provide MPTCP support for the MPTCP-incapable hosts.

The basic idea of the MPTCP proxy was first put forward in the initial IETF draft [6], where Klein *et al.* defined two types of proxies, implicit proxy and explicit proxy. An implicit proxy resides on the direct routing path between two hosts and is transparent to them. This allows hosts to establish the connection directly with each other, while the proxy can obtain all information via packet inspection, insert and modify packets as necessary and thereby create the MPTCP-TCP split connection. In contrast, an explicit proxy does not reside on the path used for connection initiation; explicit signaling is required to introduce the proxy to the connection. Subsequent studies are mainly based on this work, and more detailed designs are made for different kinds of proxy.

Some works are focused on implicit proxies. In [7], Detal *et al.* developed an MPTCP-TCP protocol converter. The converter acts like an implic-

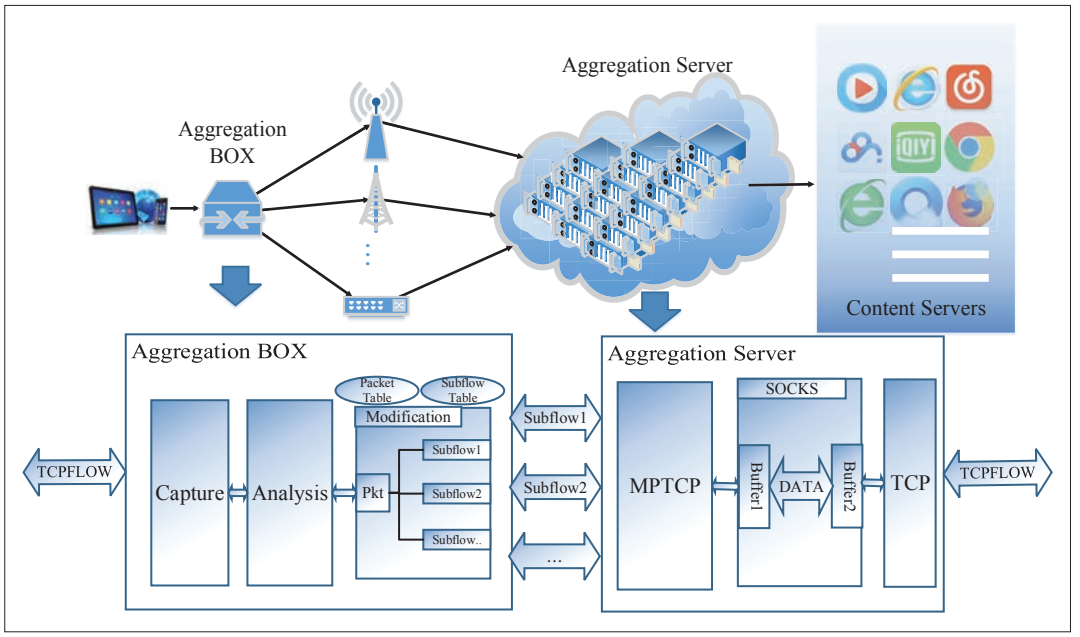

FIGURE 2. Framework of T-MPTCP.

it proxy and requires a new MPTCP option to be introduced, which will allow hosts to announce the destination server's address to the proxy. A more general splitter/combiner proxy architecture for regular TCP was reported in [8]. However, these two mechanisms are difficult to extend and deploy because they need to largely modify the kernel. Another line of work is focused on explicit proxy. The work in [10] lets the MPTCP-capable users (e.g., mobile devices) contact with MPTCP-incapable servers (e.g., application services) through MPTCP. The authors proposed a framework based on an explicit proxy to observe real smartphone applications over MPTCP. In this framework, a MPTCP-capable smartphone downloaded data from the traditional server through a MPTCP-enabled SOCKS proxy. In [12], the method of MPTCP proxy was theoretically analyzed and its practicality was tested.

In these schemes, only one proxy is placed close to either the client or the server when one of the communication peers is MPTCP-incapable. However, there may be a situation where both of the communication peers are MPTCP-incapable. We have first proposed TMPP for both of the two MPTCP-incapable hosts in [14]. TMPP uses two proxies at both the user and server side, and both the proxies need to modify the received packets and transmit them again. Nevertheless, TMPP is difficult to implement as it makes many changes to the kernel and both the proxies need to be on the path from the user to the content server. In [15], Olivier *et al*. also provided a two proxies solution which can be created within a single round-trip time. However, both the solutions use implicit proxies, which can only be deployed on the original routing path and have less flexibility to gain extra benefit from different proxy locations.

Framework of T-MPTCP

The Main Design Motivation

T-MPTCP is a new multipath deployment scheme to achieve the advantages of MPTCP for MPTCP-incapable hosts. It requires no changes on the existing users or content servers, so it is easy to implement and deploy. Besides, T-MPTCP provides transparent multipath transmission between two TCP hosts. In this case, a TCP/MPTCP conversion is provided in the network so the MPTCP-incapable host can also enjoy the benefits of multipath transmission.

System Components

As shown in Fig. 2, the proposed system has two main components in the network: the aggregation box and the aggregation server. The aggregation box at the user side acts as an implicit proxy, capturing packets and modifying their header to transparently convert TCP flow to MPTCP flow. The aggregation server at the server side acts as an explicit proxy. Through the universal explicit proxy protocol SOCKS, it splits the transmission flow into two parts: a MPTCP flow before the aggregation server and a TCP flow after it. In this way, multipath transmission is ultimately achieved for the path between user and server. While there is no bottleneck on the user and server side, the multipath transmission in the middle of the path can accelerate the overall transmission.

The aggregation box is a lightweight device, which can be easily deployed at the user's gateway to serve local users. Due to the diversity of devices, it is difficult to have MPTCP supported for all of them, but the aggregation box at the gateway makes the problem easier. Specifically, it is equipped with several types of network interfaces and provides bandwidth aggregation for users. It is placed on the way between the users and the network and can modify packets that pass through it and send them through different interfaces. The aggregation box provides users with transparent TCP/MPTCP conversion, thus users do not need to make any changes. We use netfilter for packet handling at the kernel level, which has a linear execution speed and does not take up too much CPU resource.

The aggregation server is a virtual machine with explicit proxy function which can be deployed

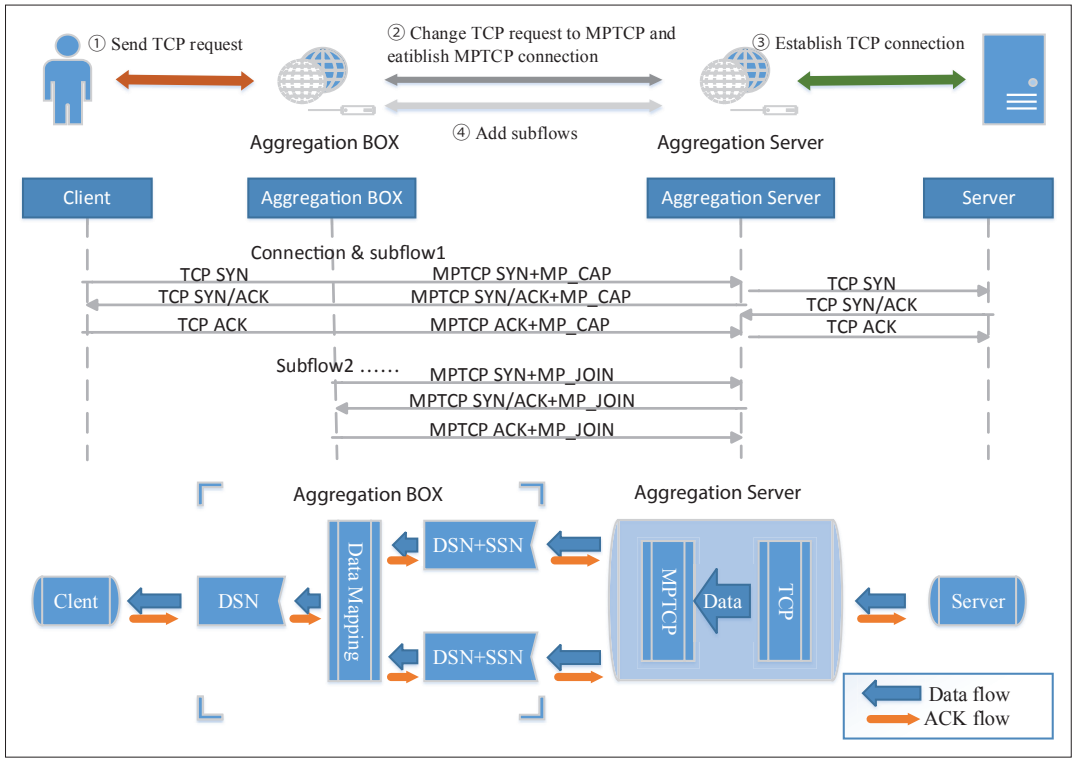

FIGURE 3. Illustration of T-MPTCP protocol.

at rented clouds to provide MPTCP accelerated service for different kinds of content servers. The explicit MPTCP proxy follows exactly the same setup as the one that was launched commercially by Korean Telecom. The aggregation server is usually not on the original transmission path from the user to the server. The user can maximize the performance of aggregated bandwidth by selecting a suitable location of the aggregation server.

Aggregation Box: The aggregation box provides the implicit proxy function for users. It performs packet capture, analysis, and modification to complete the TCP/MPTCP conversion process. As shown in Fig. 2, when a packet arrives, the aggregation box captures it and analyzes its header. If the user device supports MPTCP, the aggregation box only provides the general forwarding service, and the user directly communicates with the server using the MPTCP connection. Otherwise, the aggregation box modifies the packet header, converts TCP packets to MPTCP packets, and sends them through different interfaces to the aggregation server. It establishes and manages subflows by joining the MPTCP option and token in the TCP header and changes one TCP destination address to several MPTCP destination addresses. For data transmissions, the aggregation box maps between the corresponding sequence spaces and rewrites the TCP headers accordingly. When the packet is sent from the TCP side to the MPTCP side, regular TCP sequence numbers are mapped to MPTCP sequence numbers while the subflows use independent sequence numbers. Conversely, when the packet is sent to the TCP side from the MPTCP side, the MPTCP sequence numbers are mapped to TCP sequence numbers.

Aggregation Server: The aggregation server utilizes the SOCKS protocol, which is one of the most widely used proxy protocols to provide MPTCP explicit proxy function for servers. It splits the whole connection into two parts: one is between the user and the aggregation server, and the other is between the aggregation server and the content server. The MPTCP flow from the user modified by the aggregation box is cut down at the aggregation server and forms a MPTCP connection. Another regular TCP connection is established between the aggregation server and the content server. In order to ensure the data transmission between user and server, data is copied between the TCP's buffer and the MPTCP's buffer inside the aggregation server. Note that utilizing SOCKS may add extra delay due to the three-way handshake at the beginning of connection establishment, while it incurs little to no delay during the transmission. However, T-MPTCP is more likely proposed for large data transmission with bandwidth aggregation service, so the extra delay at the beginning can be ignored.

Details of the T-MPTCP Protocol

The communication process of T-MPTCP, which provides MPTCP transport service for two MPTCP-incapable hosts, is shown in Fig. 3. The whole process mainly includes two phases: connection establishment and data transmission. The user makes the request first to establish an MPTCP connection between the user and server through two MPTCP proxies. Then, the server can transfer data using multiple paths in the network simultaneously.

Connection Establishment

As shown in Fig. 3, the connection establishment process consists of four main steps:

- The client sends a TCP request to the aggregation server. It first sends a TCP SYN packet to the aggregation server. Then it will receive

a TCP SYN/ACK packet and send back a TCP ACK packet. After three-way handshakes, for the client, it establishes a TCP connection.

- The aggregation box modifies the TCP request to a MPTCP proxy request by rewriting the TCP header and sending it to the aggregation server. After three-way handshakes, for the aggregation server, it establishes a MPTCP connection.
- The aggregation server sends a TCP request to the content server and establishes a regular TCP connection between them after three-way handshakes.
- The aggregation box initiates a request to the Aggregation Server for adding subflow. Through the information remaining at the aggregation box, it sends MPTCP SYN packets with the MP_JOIN option to the aggregation server. After three-way handshakes, a new subflow is added to the MPTCP connection.

In so doing, multiple subflows are established between the aggregation box and the aggregation server to aggregate bandwidth for users.

Data Transmission

After the connection is established, the server will transmit data to the user. In fact, the aggregation server splits the whole connection into two single connections. It first receives data from the server and stores them, then sends them to the client. Thus, the data transmission consists of two stages:

- Data is first sent to the aggregation server through a complete TCP connection. The aggregation server receives the data and sends ACK back to the server, as with conventional TCP.
- After data arrives at the aggregation server, the aggregation server copies data through TCP to the MPTCP connection and transmits it from multiple interfaces to the aggregation box. The aggregation box captures the packet and modifies its header. Different destination addresses will be changed into one single address of the user, and the MPTCP sequence number (which includes DSN and SSN) will be mapped into a TCP sequence (which only includes DSN). The aggregation box will record this mapping information for returning ACKs subsequently. When the client receives packets, it returns ACKs to the aggregation server, which will pass through the aggregation box. When ACKs arrive at the aggregation box, the aggregation box captures them and modifies their headers. The TCP sequence number will be mapped into a MPTCP sequence, and one destination address will be modified into different destination addresses according to the mapping information that the aggregation box reserved. Then the aggregation box forwards the packet through the corresponding interface to the aggregation server.

Proxy Location Selection

In our design of T-MPTCP, aggregation servers use the explicit proxy mode and can be deployed in a number of different cloud platforms in different places. Due to the variation of network environments and user locations, the choice of an aggregation server can impact the aggregation effect.

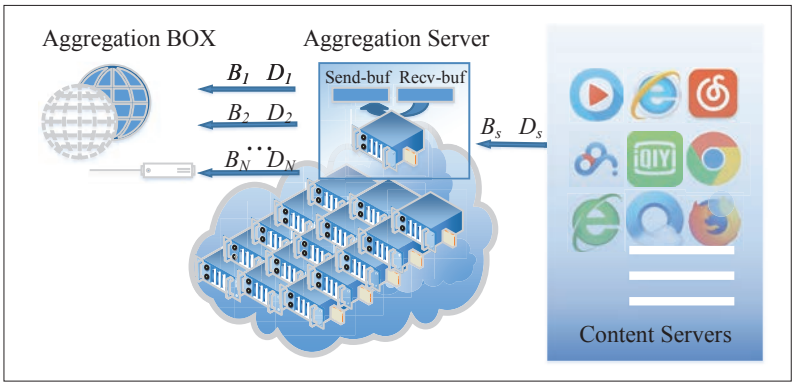

FIGURE 4. System model of PLS algorithm.

Naive strategies such as random selection likely will not give good aggregation performance. The reason is that effective throughput experienced by the user is affected by the path conditions of the aggregation servers. Here, we develop a Proxy Location Selection (PLS) mechanism for our proposed T-MPTCP scheme to improve the performance. Specifically, the PLS algorithm is deployed at the aggregation box. It periodically measures the available bandwidth and delay of each aggregation server, and then chooses the best proxy location of the aggregation server. When the aggregation box receives a TCP request from a user, it changes the TCP request to a MPTCP proxy request and sends it to the selected aggregation server. The PLS mechanism is simple yet efficient, and it can be naturally integrated in our proposed framework.

As shown in Fig. 4, the T-MPTCP architecture includes one aggregation box at the user's home gateway, and several aggregation servers running on cloud servers at different places. Each aggregation server is a candidate server and its information is stored at the aggregation box. An aggregation server establishes one TCP flow to the content server while sending multiple TCP subflows to the aggregation box. The paths to different aggregation servers have different available bandwidths and delays. The PLS mechanism thus runs the following steps to select a suitable aggregation server for the user:

- The aggregation box periodically sends probe packets to each aggregation server through all available paths between the aggregation box and aggregation servers and detects their conditions. Assume there are N paths to a candidate aggregation server, the aggregation box obtains (B_1, D_1), ..., (B_N, D_N) of each path after probing, where B_i and D_i are the bandwidth and delay of path i, respectively.
- When an aggregation server receives the probe packet from the aggregation box, it further sends a probe packet to the content server and detects the path condition from the aggregation server to the content server. After that, it sends the path condition (B_s, D_s) back to the aggregation box, where B_s and D_s are the bandwidth and delay of path between aggregation server and content server, respectively.
- For each candidate aggregation server, the aggregation box updates and stores its path conditions $<(B_1, D_1), ..., (B_N, D_N), (B_s, D_s)>$ after a periodic probe is completed. When a user needs to connect to the server, the aggregation

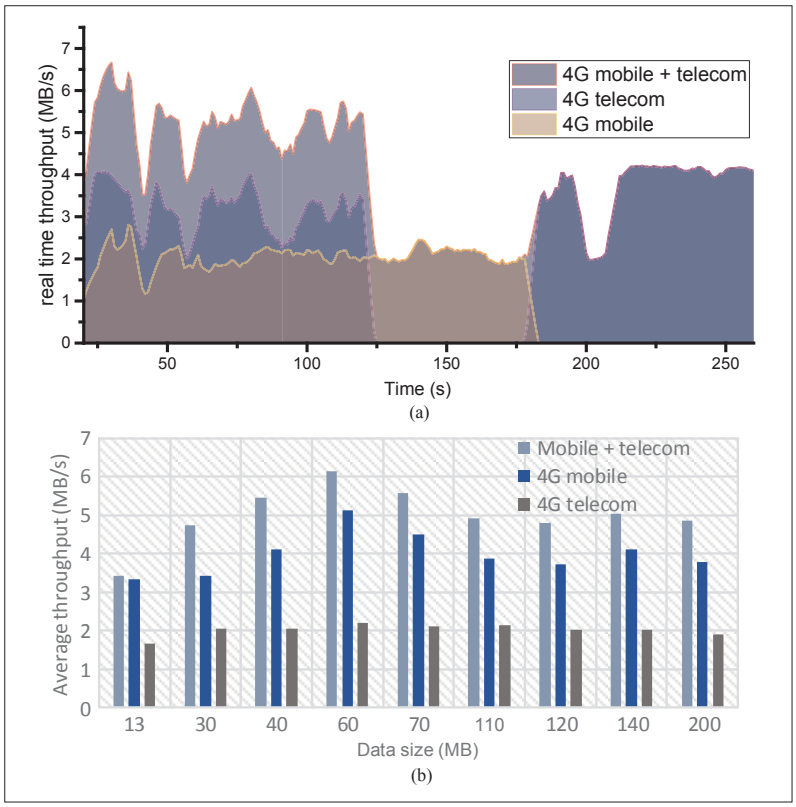

FIGURE 5. Performance evaluation: a) real throughput; b) average throughput.

box uses the collected information to select the best aggregation server. For applications such as file downloading and video streaming that need high throughput, the aggregation server that provides the maximum aggregation bandwidth should be chosen, which means that its aggregation bandwidth of B_1, ..., B_N and B_s has the maximum value.

The periodic detection process of PLS is an active detection process and the probe interval can be set according to the network environment. When the network environment is relatively dynamic, the time interval can be set appropriately smaller. Otherwise, a longer interval can be set to reduce the additional overhead of periodic detection. In practical applications, users can set the time interval according to their needs. In addition, the periodic detection function can be turned off during idle time, such as at night, to reduce energy consumption.

Performance Evaluation

Our proposed T-MPTCP has two components: the aggregation box at the user side and the aggregation server at the server side. The aggregation box is a lightweight device, which can replace gateway functionality and achieve multipath transmissions. The aggregation server is implemented at several clouds in different places, such as Ali cloud and Amazon EC2. We begin with the evaluation of the effectiveness of bandwidth aggregation of T-MPTCP, and then evaluate the performance of the PLS algorithm.

We put an aggregation box in the laboratory, which is equipped with 4G telecom and 4G mobile. The test host and aggregation box are connected using a wired link, and data from the external network can be obtained simultaneously through the two interfaces of the aggregation box. We also rent several virtual machines on Ali cloud and Amazon EC2 to implement aggregation servers.

In the experiment, we use a Linux machine as the user and download the data from a server on a Chrome browser. All experiments were completed within the same time frame but the network fluctuation is inevitable. In other words, it cannot guarantee that the network bandwidth is always constant.

Evaluation of Bandwidth Aggregation

Figure 5a shows the performance of simultaneous data transmissions through multiple interfaces using T-MPTCP. The whole data transmission process is divided into three phases. In the first phase, we activate the function of proxy and transmit data through both interfaces simultaneously. In the latter two phases, we turn off the function of proxy and use only one interface of the aggregation box to transfer data. We use Wireshark (https://www.wireshark.org/) to measure the real time throughput and calculate the average throughput every 10 seconds. From the figure we can see that the simultaneous use of two interfaces can effectively aggregate the bandwidth of two interfaces, so as to obtain better transmission performance. When using a single interface, the average transmission rate of 4G mobile and 4G telecom are 3.7 MB/s and 1.8 MB/s, respectively. When both interfaces are used together, although the high data rate path (4G mobile) is slowed down to 3.1 MB/s, the overall throughput reaches 4.8 Mb/s, which is more than only using one path.

Meanwhile, we use two interfaces to download content from the same content server using regular TCP respectively. Then we repeat the same measurement using T-MPTCP, which achieves the purpose of bandwidth aggregation. In each situation, we calculate the average throughput of different data sizes.

As shown in Fig. 5b, it can be seen that the average throughput is significantly improved when using T-MPTCP. Among them, 4G telecom's transmission rate is always stable for data of different sizes, while the transmission rate has a little fluctuation for different data sizes in 4G mobile. The aggregation bandwidth also changes with the fluctuation of 4G mobile, which indicates that the T-MPTCP also has a good polymerization effect on the flow of different data rates. The effect of aggregate bandwidth is not obvious when the download data size is too small, because of the overhead in selecting the proxy and creating subflows. Comparing with only using one interface at a time, T-MPTCP aggregates bandwidth of different interfaces and provides the highest throughput in any case. In our test, T-MPTCP increases the throughput up to 40 percent relative to only using 4G telecom and up to 180 percent relative to only using 4G mobile.

Evaluation of the PLS Algorithm

We also rent several virtual machines on Ali cloud and Amazon EC2 at different places so as to verify that the MPTCP PLS algorithm can find the optimized proxy thus to improve performance. As expected, both the delay and bandwidth of each path between the aggregation box and the aggregation server, and between the aggregation server and the content server, affect the aggrega-

tion server, we measure the delay and remaining bandwidth of each path in the connection, and also the average data rate of downloading a 100 MB data from the content server through the MPTCP proxy in the same time.

Table 1 shows the measurement results by selecting different proxy locations. We implement aggregation servers in five different places, and the index represents the cloud location. Also, we give the output of the PLS algorithm. Table 1 shows the aggregation effect w.r.t. the aggregation servers provided by different cloud service providers; the highlighted columns represent the selected aggregation server using the PLS algorithm. Throughput is affected by the bandwidth and delay of paths between end hosts. Using the PLS algorithm, we can effectively select a suitable location of the aggregation server and improve the overall performance.

Proxy	1	2	3(selected)	4	5
B_1/Mbps	21.89	22.63	27.57	1.38	5.1
D_1/ms	53.37	44.60	53.97	43.67	113.15
B_2/Mbps	13.45	15.38	19.75	13.91	12.9
D_2/ms	91.745	64.34	67.66	57.45	179.58
B_s/Mbps	92.8	98.4	56.31	36.31	63.28
D_s/ms	42.64	36.05	27.50	38.19	36.9
Throughput (Mbps)	24.38	28.45	36.89	13.03	17.02

TABLE 1. Measurement result of PLS algorithm.

Conclusion

In this article, we proposed T-MPTCP, a deployable MPTCP-based bandwidth aggregation system without any modifications to the existing network architecture. We also designed a MPTCP Proxy Location Selection algorithm to choose the optimal placement of aggregation servers in the cloud platform. Our design was implemented and evaluated in a testbed, which showed improved performance in terms of aggregated throughput.

Acknowledgment

This work is supported in part by the Youth Innovation Promotion Association of the Chinese Academy of Sciences (CAS) under Grant No. Y202093, and the National Natural Science Foundation of China under Grant No. 61972371.

References

[1] H. Sinky, B. Hamdaoui, and M. Guizani, "Seamless Handoffs in Wireless HetNets: Transport-Layer Challenges and Multi-Path TCP Solutions with Cross-Layer Awareness," *IEEE Network*, vol. 33, no. 2, 2019, pp. 195–201.

[2] L. Li *et al.*, "A Measurement Study on Multi-Path TCP with Multiple Cellular Carriers on High Speed Rails," *Proc. 2018 Conf. ACM Special Interest Group on Data Communication (SIGCOMM)*, ACM, 2018, pp. 161–75.

[3] K. Xue *et al.*, "DPSAF: Forward Prediction Based Dynamic Packet Scheduling and Adjusting with Feedback for Multipath TCP in Lossy Heterogeneous Networks," *IEEE Trans. Vehicular Technology*, vol. 67, no. 2, 2018, pp. 1521–34.

[4] Q. Peng *et al.*, "Multipath TCP: Analysis, Design, and Implementation," *IEEE/ACM Trans. Networking*, vol. 24, no. 1, 2016, pp. 596–609.

[5] C. Raiciu *et al.*, "How Hard Can It Be? Designing and Implementing a Deployable Multipath TCP," *Proc. 9th USENIX Conf. Networked Systems Design and Implementation (NSDI)*, Proc. USENIX, 2012, pp. 399–412.

[6] T. Klein and G. Hampel, "MPTCP Proxies and Anchors," 2012, draft-hampel-mptcp-proxies-anchors-00, IETF; accessed on May., 2021; available: https://tools.ietf.org/html/draft-hampel-mptcp-proxies-anchors-00.

[7] G. Detal, C. Paasch, and O. Bonaventure, "Multipath in the Middle (Box)," *Proc. 2013 Workshop on Hot topics in Middleboxes and Network Function Virtualization (HotMiddlebox)*, ACM, 2013, pp. 1–6.

[8] T. Ayar *et al.*, "TCP over Multiple Paths Revisited: Towards Transparent Proxy Solutions," *Proc. 2012 IEEE Int'l. Conf. Commun. (ICC)*, IEEE, 2012, pp. 109–14.

[9] G. Hampel, A. Rana, and T. Klein, "Seamless TCP Mobility Using Lightweight MPTCP Proxy," *Proc. 11th ACM Int'l. Symposium on Mobility Management and Wireless Access (MobiWac)*, ACM, 2013, pp. 139–46.

[10] Q. De Coninck *et al.*, "Observing Real Smartphone Applications over Multipath TCP," *IEEE Commun. Mag.*, vol. 54, no. 3, 2016, pp. 88–93.

[11] O. Bonaventure *et al.*, "0-RTT TCP convert protocol," 2020, RFC 8803, IETF; accessed in May, 2021; available: https://www.ietf.org/rfc/rfc8803.txt.

[12] S. Chung, S. Moon, and S. Kim, "The Virtualized MPTCP Proxy Performance in Cellular Network," *Proc. 9th Int'l. Conf. Ubiquitous and Future Networks (ICUFN)*, IEEE, 2017, pp. 703–07.

[13] B. Chen *et al.*, "Modeling and Analysis of MPTCP Proxy-Based LTE-WLAN Path Aggregation," *Proc. 2017 IEEE Global Commun. Conf. (GLOBECOM)*, IEEE, 2017, pp. 1–7.

[14] K. Xue *et al.*, "TMPP for Both Two MPTCP-Unaware Hosts," 2013, draft-xue-mptcp-tmpp-unwarehosts- 02, IETF; accessed in May, 2021; available: https://tools.ietf.org/html/draft-xue-mptcp-tmpp-unware-hosts-02.

[15] O. Bonaventure and S. Seo, "Multipath TCP Deployments," *IETF J.*, vol. 12, no. 2, 2016, pp. 24–27.

Biographies

Jiangping Han (jphang@mail.ustc.edu.cn) received her B.S. degree from the Department of Electronic Engineering and Information Science (EEIS), University of Science and Technology of China (USTC), in 2016. She is currently working toward the Ph.D. degree in the Department of EEIS, USTC. Her research interests include future Internet architecture and transmission optimization.

Kaiping Xue [M'09, SM'15] (kpxue@ustc.edu.cn) received his bachelor's degree from the Department of Information Security, University of Science and Technology of China (USTC), in 2003, and received his Ph.D. degree from the Department of Electronic Engineering and Information Science (EEIS), USTC, in 2007. Currently, he is a professor in the School of Cyber Security and Department of EEIS, USTC. His research interests include next-generation Internet architecture design, transmission optimization and network security. He serves on the Editorial Board of several journals, including the *IEEE Transactions on Dependable and Secure Computing* (TDSC), the *IEEE Transactions on Wireless Communications* (TWC), and the *IEEE Transactions on Network and Service Management* (TNSM). He has also served as a (Lead) Guest Editor for many respected journals/magazines, including IEEE JSAC, *IEEE Communications Magazine*, and *IEEE Network*. He is an IET Fellow and IEEE Senior Member.

Wenjia Wei (wwj2014@mail.ustc.edu.cn) received the B.S. degree from the School of Information Science and Engineering in 2013, and received his Ph.D. degree in information and communication engineering from the Department of Electronic Engineering and Information Science (EEIS), University of Science and Technology of China (USTC), in 2020. His research interests include future Internet architecture design and transmission optimization.

Yitao Xing (ytxing@mail.ustc.edu.cn) received his B.S. degree in information security from the School of the Gifted Young, University of Science and Technology of China (USTC), in 2018. He is currently working toward the Ph.D. degree in the School of Cyber Security, USTC. His research interests include future Internet architecture and transmission optimization.

Jianqing Liu (jianqing.liu@uah.edu) received his B.Eng. degree from the University of Electronic Science and Technology of China, Chengdu, China, in 2013, and the Ph.D. degree from the Department of Electrical and Computer Engineering, University of Florida, Gainesville, FL, USA, in 2018. Currently, he is an assistant professor in the Department of Electrical and Computer Engineering at the University of Alabama in Huntsville. His research interests include wireless networking and network security in cyber-physical systems.

Peilin Hong (plhong@ustc.edu.cn) received her B.S. and M.S. degrees from the Department of EEIS, USTC, in 1983 and 1986, respectively. Currently, she is a professor in the Department of EEIS, USTC. Her research interests include next-generation Internet, policy control, IP QoS, and information security.