MPTCP Meets Big Data: Customizing Transmission Strategy for Various Data Flows

Yitao Xing, Jiangping Han, Kaiping Xue, Jianqing Liu, Miao Pan, and Peilin Hong

Abstract

In recent years, network transmission carries big data of various types and sizes, which often have different transmission requirements. Transmission of data brings challenges to traditional transport layer protocols; meanwhile, performance improvements at endpoints and networks provide opportunities for new transport protocol design. Among the new designed transport layer protocols, multipath TCP is a promising one that provides better performance with multiple paths transmitting data simultaneously. However, when dealing with different types and sizes of data, MPTCP does not always work well, as all data streams are processed based on the same transmission strategy, which is simple but not always optimal. To fill this gap, in this article, we first give optimal strategies for different data flows, and propose a cross-layer solution for MPTCP, named FSA-MPTCP, to automatically provide flexible and optimal transmission strategies for different data flows. Finally, we conduct some experiments, and the results show that FSA-MPTCP adaptively provides better performance for different kinds of flows.

Introduction

With the emergence of new network applications and technologies, the network becomes increasingly complex, and more diverse demands need to be satisfied. Under the explosive increase of global data, the term "big data" is mainly used to describe enormous datasets [1, 2], and is usually described with a "3Vs" model, that is, the increase of volume, velocity, and variety of data [3]. Transmission of big data brings challenges to transport layer protocol design, which often have performance requirements of higher throughput, better robustness, and greater flexibility.

Among the current transport layer protocols, multipath TCP (MPTCP) is the most competitive one, which is designed to provide robust data transmission and high efficiency with multiple subflows working together [4]. Unlike regular TCP, MPTCP establishes multiple TCP connections on different interfaces under one MPTCP connection, which are called subflows. These subflows can effectively utilize multi-interface network resources. Compared to TCP, MPTCP has potential for performance improvement (e,g,, bandwidth aggregation, robustness increasing, and load balancing). As illustrated in Fig. 1, there are many network applications that can benefit from MPTCP.

Until now, many studies have been conducted to improve the transmission performance of MPTCP, for example, on coupled congestion control and data scheduling [5, 6]. However, the existing methods are mainly designed to improve the performance of MPTCP for general circumstances, that is, how to effectively transfer a continuous stream of upper-layer data to the receiver. In fact, the constantly changing congestion window and asymmetric network paths of MPTCP subflows make it necessary to handle data of different sizes with different strategies:

- Elephant flows with large data volume need to be efficiently transmitted through all available paths for high bandwidth, which can use round-robin or default (low round-trip time, RTT) schedulers.
- Mice flows with small data volume, which carry control signals and short messages, prefer high robustness and low latency.

These kinds of flows need redundancy-based schedulers and should be completely transferred in as few scheduling cycles as possible [7, 8]. The last kind of flow, the mosquito flow, which is a tiny flow we define later, can be transmitted only on the best path.

There have been many works to optimize MPTCP for these different scenarios. However, the current MPTCP with inflexible transmission strategies cannot meet the performance requirements of different data flows properly. The reason is that current MPTCP is unaware of data volume to be sent, so it handles all kinds of data flows with the same transmission strategy, which will lead to poor performance when the strategy is not suitable for the specific kind of flow. Therefore, in the context of big data transmission, making MPTCP more flexible for different data flows can bring performance improvement. The big-data-driven approach provides an opportunity in this case to use previous data to estimate data to be sent in the future. Thus, we can predict and get extra attribute information about sending data. With such information, to make MPTCP aware of the data information and provide flexible transmission strategies for different data flows, a cross-layer design can potentially be a good solution.

In the first section of this article, by conducting experiments for data transmission with MPTCP, we find out different optimal strategies for flows with different sizes. Then we propose flow-size-aware MPTCP (FSA-MPTCP), a cross-layer design for MPTCP, which is able to receive and analyze the size information of data flows from an upper layer,

Yitao Xing, Jiangping Han, Kaiping Xue (corresponding author), and Peilin Hong are with the University of Science and Technology of China; Jianqing Liu is with the University of Alabama; Miao Pan is with the University of Houston.

Digital Object Identifier: 10.1109/MNET.011.1900438

0890-8044/20/$25.00 © 2020 IEEE

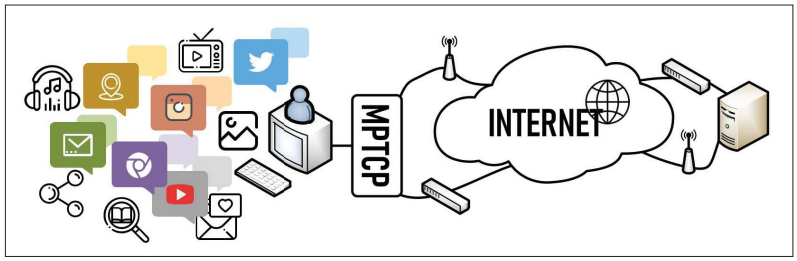

FIGURE 1. Big data and MPTCP.

and provides customized transmission strategies for different kinds of flows. FSA-MPTCP can transmit data with the optimal strategy and guarantee diverse quality of service (QoS) automatically, and outperforms regular MPTCP when dealing with a wide variety of data flows in big data. We also conduct several experiments to show that FSA-MPTCP provides better performance and greater flexibility for various data flows in big data scenarios.

The rest of this article is organized as follows. The following section gives background for our work. We then explore optimal transmission strategies for different data flows. Finally, we propose FSA-MPTCP and present the evaluation. Further discussion and a conclusion are given in the final sections.

Background

The current network is full of big data of different types and sizes, which makes data transmission a big challenge [1]. To tackle the challenge, MPTCP, a promising transport layer protocol, is proposed to meet the requirements of higher throughput, better robustness, and greater flexibility. In this section, we review the background of big data and MPTCP.

Big Data

Under the explosive increase of global data, the term "big data" is mainly used to describe enormous datasets. Unlike traditional datasets, big data includes masses of unstructured data that needs more advanced tools to analyze. There are several representative applications of big data, including the Internet of Things (IoT), online social networks, medial applications, collective intelligence, smart grid, and so on, which generate various kinds of data. In general, the characteristics of big data are described with the above-mentioned 3Vs model:

Volume: The scale of dataset becomes increasingly larger.

Velocity: Data collection and analysis should be much more rapidly and opportunely conducted.

Variety: The data is of various types, which include not only traditional structured data, but also semi-structured and unstructured data including video, audio, plain text, webpages, and so on.

Big data brings both opportunities and challenges for network performance improvement [3]. On one hand, based on big-data-driven technologies, the network can be more functional and provide better QoS for users [9, 10]. On the other hand, the large volume of highly unstructured and various big data requires better transmission performance of the network, challenging the transport layer protocols to support various types of data flows more flexibly.

MPTCP

MPTCP is one of the most popular multipath transmission protocols [11]. As an extension of TCP, it provides multipath transmission with little network modification. MPTCP allows a single data stream to be split across multiple paths, and manages subflows with functions such as scheduler, congestion controller, and path manager [5, 6, 12], which may lead to different transmission performance.

Scheduler: There are three schedulers introduced in the MPTCP Linux kernel [13], which are minRTT as the default scheduler, round-robin scheduler, and redundant scheduler. Different schedulers provide different transmission performance. For example, the round-robin scheduler sends a packet on each subflow in turn to simply aggregate bandwidth, while the minRTT scheduler sends packets on the subflow with the lowest RTT with priority to avoid harmful effects of slower flows. The redundant scheduler, just as its name implies, sends the same redundant data packets on each subflow against packet loss and provides better robustness.

Congestion Controller: MPTCP congestion control algorithms inherit from the original TCP congestion algorithm, such as CUBIC, but additionally aim at fairness, load balancing, and data migration. Also, most congestion control algorithms include a slow start phase and a congestion avoidance phase, which normally uses the additive-increase multiplicative-decrease (AIMD) algorithm. In both phases, congestion windows increase every RTT. In other words, RTT influences congestion windows as well as the throughput of TCP and MPTCP connections.

Path Manager: MPTCP transmits data through all the paths simultaneously to achieve higher throughput by default, and it also provides an optional backup mode to save network traffic or to avoid harmful effects from one or more paths in poor condition. In backup mode, paths that are specified as backup will not be used unless the default path is unavailable.

MPTCP is a promising transport protocol for big data. As shown in Fig. 1, a large number of network applications can benefit from MPTCP transmission. For example, the aggregated bandwidth provided by MPTCP efficiently transports the large volume of big data such as video chunks from YouTube with high velocity. MPTCP offers several schedulers and path managers that may benefit a variety of data flows in big data. In the next section, we analyze and discover the optimal MPTCP transmission strategies for flows of different sizes.

Optimal MPTCP Transmission Strategies for Different Sizes of Flows

Big data challenges the network for a better transport layer protocol to transmit all kinds of data flows. MPTCP is a promising protocol that has the potential to tackle the challenge. In this section, we try to find out what the optimal MPTCP transmission strategies are for different kinds of data flows using experiments.

The critical factor that influences the performance of MPTCP is different according to

the sizes of the flows [8]. In previous studies, data flows are often divided into two categories by size: elephant flows and mice flows. For elephant flows, the critical factor is bandwidth, while for mice flows, it should be latency [14]. However, MPTCP makes the situation more complicated because data is transmitted through multiple paths of various conditions at the same time, which may lead to results that differ from regular TCP. And to our knowledge, there is a lack of study on how the different sizes of flows interact with different characteristics of paths in MPTCP. Thus, we conduct some experiments to find out optimal strategies for different sizes of flows.

Experiments and Analysis

To find out the optimal transmission strategies for different sizes of flows in MPTCP, we conduct the following two experiments.

In the first experiment, we download different files from the same server with two subflows running on paths that have the same bandwidth of 5 Mb/s but different RTT of 40 ms and 100 ms, respectively. Moreover, the first subflow is established over paths of different RTT. That is, one server establishes its initial subflow over a path of 40 ms RTT, and the other one starts with a path of 100 ms RTT. Figure 2a shows that the server which establishes the initial subflow of 40 ms RTT provides much shorter flow completion time than that of 100 ms RTT — a "fast" initial subflow always shortens the flow completion time by about 100 ms.

The reason the fast initial subflow provides better performance is that a non-negligible part of the smaller flows is sent through the first subflow before other subflows are established, and the subflow with lower RTT achieves higher throughput. For larger flows, the part of data that is sent through the initial subflow does not increase and covers a small portion of all data; thus, the improvement of fast initial subflow is slighter. In a word, it is a good strategy to establish the initial subflow over the path with lowest RTT, especially for smaller data flows.

In addition, in data flows of only 4 kB, the first subflow carries all the data traffic, leaving nothing for other subflows that are established later. We define these tiny flows as "mosquito flows" in our article and treat them differently.

The second experiment shows how current MPTCP schedulers work with various sizes of flows. We download different sizes of files from servers with the same path conditions but distinct schedulers. There are two subflows within one MPTCP connection, and both of them run on paths of 40 ms RTT and 5 Mb/s bandwidth and suffer from packet loss. The flow completion time is shown in Fig. 2b. As we can see from the figure, the round-robin scheduler achieves its goal to aggregate bandwidth for larger data flows such as 512 kB and 1 MB files and to achieve shorter flow completion time. For smaller flows, it provides unstable flow completion time. The redundant scheduler, however, is more stable in all test scenarios, but makes the flow completion time much longer for larger flows.

In real networks, data flows suffer from packet loss. Since there are not enough packets in a small flow to trigger fast retransmission, a single packet loss may cause retransmission time-out (RTO), leading to unbearably long flow completion time, as shown in Fig. 3. In this case, sending redundant packets on different subflows could partly avoid RTO. On the other hand, it should be noted that large flows need aggregated bandwidth for better performance, as shown in Fig. 4; thus, the current MPTCP redundant scheduler is not optimal for all flows. In a word,

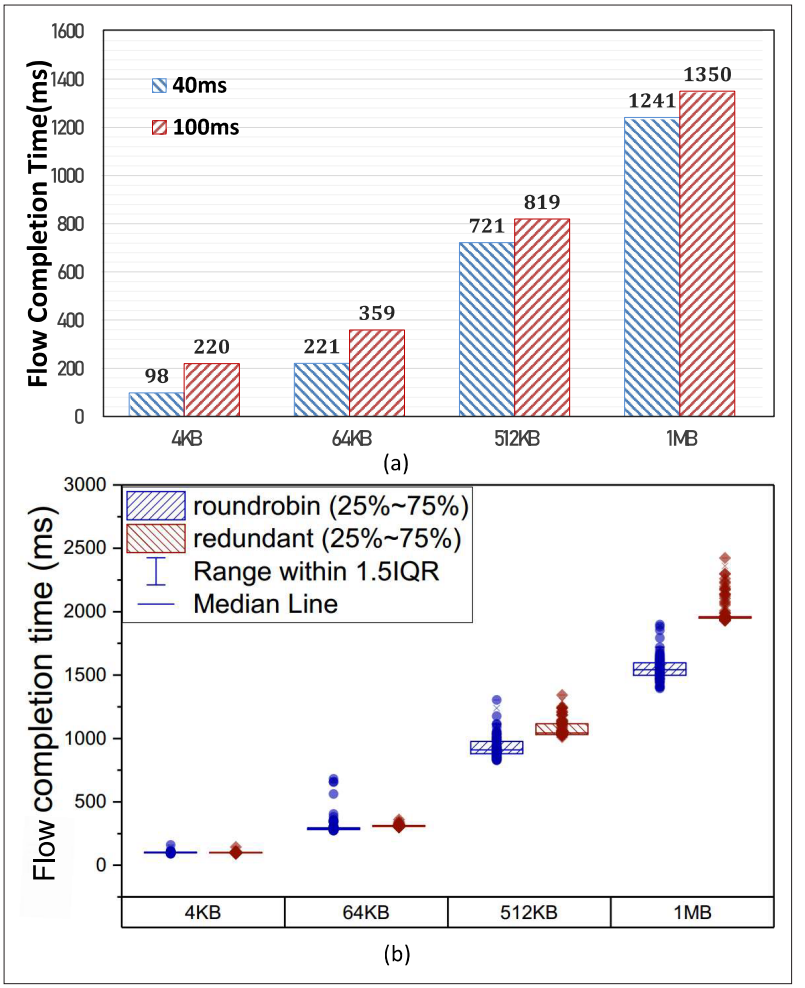

FIGURE 2. Flow completion time of different sizes of data flows with different transmission strategies.

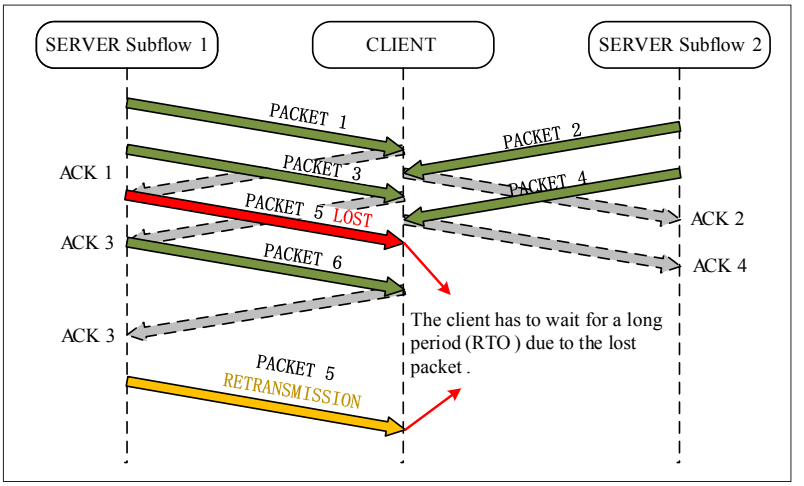

FIGURE 3. Packet loss causes RTO.

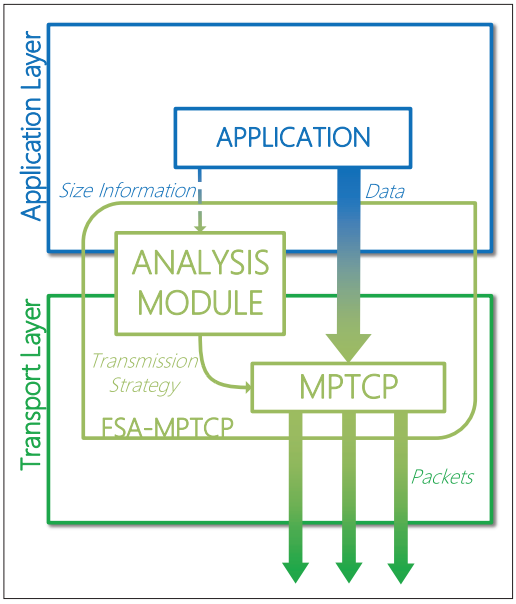

FIGURE 4. Framework of our cross-layer design for flow-size-aware MPTCP.

neither a round-robin scheduler nor a redundant scheduler can provide satisfactory transmissions for all kinds of flows, which means the optimal transmission strategies should be decided according to the characteristics of data flows.

Transmission Strategy for Different Flows

In this article, data flows are classified into three categories, namely, mosquito flows, mice flows, and elephant flows. Based on our experiments and analysis, each category needs its own transmission strategy as follows. Note that the exact definitions of the following three types of data flows are given below, and here we focus on transmission strategies qualitatively.

For Mosquito Flows: The first established subflow is so important in carrying all the data traffic that other subflows do not even need to be established. Therefore, MPTCP should choose the best path from all available ones to establish the initial subflow over it. Since other subflows carry little data traffic, MPTCP does not establish them for mosquito flows to save overhead.

For Mice Flows: The performance of mice flows is influenced mainly by RTT of subflows as well as the establishing sequence of subflows. In a real network, mice flows (e.g., control signals) are delay-sensitive, which means robustness plays an important role in mice flow transmissions. As a result, sending redundant packets is a good choice to avoid packet loss and provide better robustness for smaller mice flows. For larger ones, more new packets are necessary to improve the performance.

For Elephant Flows: The performance of elephant flows is influenced mainly by total bandwidth of all available paths. In a real network, elephant flows include bulks of flows including big files, videos, and so on. Transmission of this kind of flow needs high bandwidth to shorten the flow completion time. Therefore, when dealing with elephant flows, MPTCP should use all available subflows to transfer new packets.

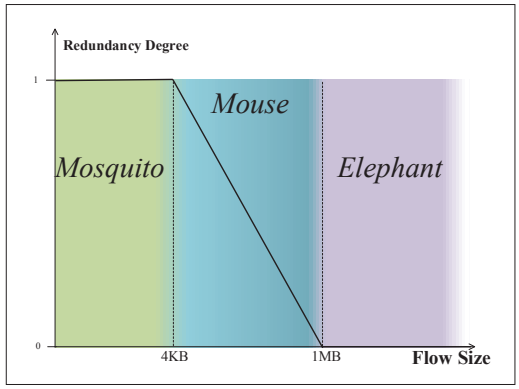

FIGURE 5. Redundancy degree α and flow classification in FSA-MPTCP

Cross-Layer Solution: Flow-Size-Aware MPTCP

Although we learn optimal MPTCP transmission strategies for various kinds of data flows, MPTCP is still unable to transport data flows flexibly with appropriate transmission strategies in the absence of the information about the data to send. In this section, we propose a cross-layer design for MPTCP called flow-size-aware MPTCP (FSA-MPTCP). FSA-MPTCP is a solution to adaptively provide optimal transmission strategies for various data flows in big data. With extra information about data flows from applications in an upper layer, FSA-MPTCP is able to automatically customize transmission strategies for data flows. The framework of FSA-MPTCP is shown in Fig. 5.

Interaction with Applications

There are many applications of big data in various aspects, including enterprises, IoT, online social networks, smart grid, video, and so on, which generate a wide variety of data. For example, big data of online social network service (SNS) mainly includes instant messages, micro blogs, shared space, and so on; while video servers generate bulks of video chunks that are much larger. FSA-MPTCP enables such servers to acquire customized transmission by providing applications an application programming interface (API) to specify the size of data flows. Applications on the upper layer know about the data to be sent, and both traditional data analysis and big data analytic methods can help refine useful information from big data. As a result, the sizes of data flows are available for applications. Once the application knows about the data size and starts to send its data using an MPTCP connection, it sends the data and the exact number of *flow_size* to FSA-MPTCP; then FSA-MPTCP will choose an optimal strategy automatically for this data flow, which happens in the transport layer.

Actions in the Transport Layer

As shown in Fig. 5, an analysis module is designed between the upper layer and the transport layer. The analysis module receives the flow_size from the upper application layer, which traditional MPTCP cannot do, and classifies the flow into one of the three types, which are mosquito flow, mice flow, and elephant flow. After that, the analysis module customizes a transmission strategy

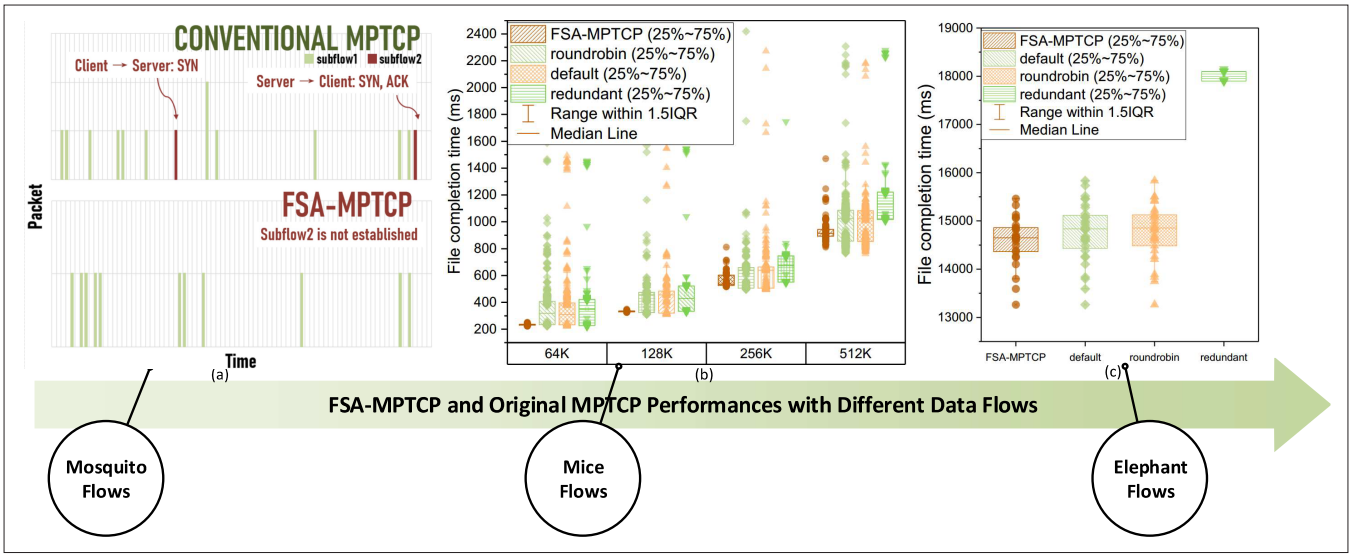

FIGURE 6. Performance evaluation for FSA-MPTCP: a) the final part of mosquito flows using original MPTCP and FSA-MPTCP; b) flow completion time of mice flow using FSA-MPTCP and original MPTCP with different schedulers; c) flow completion time of elephant flow using FSA-MPTCP and original MPTCP with different schedulers.

according to the *flow_size* of the data to be sent. Finally, MPTCP transmits the data flows following the strategy that the analysis module decides.

In the following, we carefully explain the flow classification and the fine-grained transmission strategies for different flows according to the flow_size.

Flow Classification in Analysis Module: As mentioned previously, data flows are classified into one of the three types because of their different behaviors when they are transferred using FSA-MPTCP. The three types of flows are defined as follows:

- Mosquito flows are very tiny mice flows newly defined in this article. We define data flows that are smaller than the initial value of cwnd (*IW*) as mosquito flows, which can be finished in one RTT. As specified in [15], *IW* = 3 × 1460 bytes = 4380 bytes ≈ 4 kB. Thus mosquito flows are those smaller than 4 kB.
- Mice flows are larger than 4 kB and smaller than 1 MB, whose flow completion time is strongly influenced by RTT of subflows of MPTCP connections.
- Elephant flows are larger than 1 MB, whose flow completion time is mainly influenced by the total bandwidth of MPTCP connections.

Transmission Strategies: After flow classification, the analysis module decides the transmission strategies of MPTCP for flows of each type using a "redundancy degree" α as shown in Fig. 6, which represents a probability to send redundant packets on certain subflows. Furthermore, there is a strategy that is suitable for all kinds of flows named initial subflow selection. The detailed description of the strategies are as follows.

For Mosquito Flow: FSA-MPTCP falls back to regular TCP using only one subflow to transfer data when dealing with mosquito flows. Since there is no redundant packet that needs to be sent in this situation, the redundancy degree $\alpha = 1$ for mosquito flows.

For Mice Flow: FSA-MPTCP calculates the redundancy degree $\alpha \in (0,1)$ to decide the strategy. The redundancy degree α is calculated as follows:

$$\alpha = 1 - \frac{flow\_size - 4\ \text{kB}}{1\ \text{MB} - 4\ \text{kB}},$$

where $0 < \alpha < 1$. In this situation, FSA-MPTCP always sends new packets on the subflow with the lowest RTT, while for other subflows, FSA-MPTCP sends redundant packets with probability α and sends new packets with probability 1 – α. That is, the larger the flow is, the more new packets will be sent instead of redundant ones.

For Elephant Flow: FSA-MPTCP sends new packets on all available subflows for higher bandwidth without any redundant packet. That is, for elephant flows, the redundancy degree is $\alpha = 0$. Except for the strategies mentioned above, FSA-MPTCP provides another strategy for all kinds of data flows, which is called initial subflow selection.

Initial Subflow Selection: FSA-MPTCP tries to start an MPTCP connection on the path with lowest RTT by leveraging subflow information left by previous FSA-MPTCP connections between two identical hosts. An FSA-MPTCP connection records the RTT of each available subflow. When a new FSA-MPTCP connection starts, it searches for records according to the address of the server to find a subflow with lowest RTT. If no record is available, FSA-MPTCP retreats and tries to connect to the server using traditional MPTCP. However, if previous FSA-MPTCP connections leave records about the server, it is able to establish the initial subflow over a path with lowest RTT, which shortens the flow completion time.

After customizing transmission strategies for data flows, FSA-MPTCP starts an MPTCP connection with an appointed server. Then the data transmission goes on like regular MPTCP but with optimal strategy and requires no modifications in the current network.

Performance Evaluation

We implement FSA-MPTCP in Linux kernel with MPTCP v0.94 [13], then conduct experiments to show the performance of different data flows. In the experiments, we use FSA-MPTCP and original

Taking advantage of big data, it is possible to customize more fine-grained transmission strategies according to the QoS requirements of specific data flows with both traditional data analysis and big data analytic methods.

MPTCP with different schedulers to transmit data flows of various sizes, and the results show the flexibility of FSA-MPTCP, which is able to transmit different data flows with optimal strategies.

FSA-MPTCP saves overhead for mosquito flows, which we can observe from the following experiment.

We download two tiny files from servers using FSA-MPTCP and original MPTCP, respectively, and capture the packets of the final part of the transmission shown in Fig. 4. FSA-MPTCP receives the data flows together with the size information from the upper layer, so it knows that the data flow is a mosquito flow. As a result, following the strategy decided by the analysis module, it falls back to regular TCP and saves overhead from establishment of other subflows. Because original MPTCP knows no extra information about the data, SYN packet and the corresponding SYN/ACK packet are sent by two hosts to establish a second subflow. This second subflow, a half-open connection, is a total waste of CPU and memory resources since it carries no data traffic. It seems that the overhead FSA-MPTCP saves is negligible, but when a large number of clients request sequences of tiny files, the overhead cannot be ignored.

FSA-MPTCP provides shorter flow completion time and better robustness for mice flows. To prove that, FSA-MPTCP is used to transmit mice flows and elephant flows, as well as original MPTCP with different schedulers. To simulate real network conditions, one path has 40 ms RTT and 5 Mb/s bandwidth as a "good" path, while another path has 100 ms RTT, 5 Mb/s bandwidth, and packet loss as a "poor" path. The flow completion time is shown in Figs. 6b and 6c.

For all four sizes of mice flows in Fig. 6b, FSA-MPTCP shortens the flow completion time and provides minimal jitter of transmissions compared to MPTCP using different schedulers. For smaller flows of 64 kB, the average flow completion time of FSA-MPTCP is shorter than that of MPTCP with other schedulers because of the initial subflow selection strategy in FSA-MPTCP. Most of the data traffic is sent through a "good" path, while the "poor" path carries little data traffic. Furthermore, about 94 percent of the packets sent on the "poor" path are redundant packets against packet loss, enhancing robustness of the connections. That is why original MPTCP, no matter what scheduler is used, suffers from the poor path, and transmissions are much more unstable. For larger mice flows of 512 kB, FSA-MPTCP also achieves better performance. The flow completion time of 512 kB files is more influenced by aggregated bandwidth. Thus, MPTCP with redundant scheduler has worse performance than others. However, the fact that one subflow is in a poor network condition still hinders the performance of round-robin and minRTT schedulers due to packet loss. FSA-MPTCP has a good compromise by sending 50 percent redundant packets on the "poor" path against packet loss and the other 50 percent new packets to aggregate bandwidth, which finally leads to the shortest flow completion time in this situation. Additionally, the initial subflow selection also stops the harm that an initial subflow on a "poor" path may do to the transmissions. In a word, FSA-MPTCP is always able to customize suitable transmission strategies for different kinds of flows, which is much more flexible than original MPTCP with fixed schedulers.

As for elephant flows, FSA-MPTCP schedules no redundant packet and aggregates bandwidth of all available subflows. As a result, FSA-MPTCP provides the same flow completion time as regular MPTCP with a round-robin scheduler and a minRTT scheduler as shown in Fig. 6c, outperforming the redundant scheduler. Regular MPTCP with a redundant scheduler offers stable transmission performance; however, the flows are so large that sending new packets on the "poor" path for higher throughput outweighs sending redundant packets on it, even though lost packets make the transmissions unstable.

Generally speaking, FSA-MPTCP provides transmission strategies that are suitable for different kinds of data flows, but original MPTCP with a fixed scheduler is not always satisfactory with various flows, as we can see in Fig. 6.

Further Discussion

In this article, data size is the only consideration in our novel self-adaptive cross-layer framework for MPTCP. Although our FSA-MPTCP improves the performance for different sizes of data flows successfully, there can be more enhancement if other characteristics of data flows are considered. Characteristics such as data types, structures, and priority can be taken into consideration for efficiency and feasibility in future work. For example, once applications mark their data flows as high-priority, a cross-layer designed MPTCP is able to send packets on paths of higher throughput and low latency. If data flows are marked as low-priority, paths of less throughput or higher latency may be used for congestion balancing. Taking advantage of big data, it is possible to customize more fine-grained transmission strategies according to the QoS requirements of specific data flows with both traditional data analysis and big data analytic methods.

Conclusion

In this article, we discuss how MPTCP works with big data and how to improve its performance. Conventional MPTCP cannot deal with different data flows properly because its transmission strategy is fixed. In order to provide flexible transmission strategies for various flows in big data, we propose a cross-layer solution, called FSA-MPTCP. FSA-MPTCP leverages the extra information from applications to customize transmission strategies for different data flows. Our proposal provides better performance for data of different sizes.

Acknowledgments

This work is supported in part by the National Natural Science Foundation of China under Grant No. 61972371, the Youth Innovation Promotion Association of the Chinese Academy of Sciences (CAS) under Grant No. 2016394, and the Nation-

al Key Research and Development Program of China under Grant No. 2017YFB0801702.

References

[1] M. Chen, S. Mao, and Y. Liu, "Big Data: A Survey," *Mobile Networks and Applications*, vol. 19, no. 2, 2014, pp. 171–209.
[2] E. Ahmed *et al.*, "Recent Advances and Challenges in Mobile Big Data," *IEEE Commun. Mag.*, vol. 56, no. 2, Feb. 2018, pp. 102–08.
[3] H. Fang *et al.*, "A Survey of Big Data Research," *IEEE Network*, vol. 29, no. 5, Sept./Oct. 2015, pp. 6–9.
[4] 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 Commun.*, 2018, pp. 161–75.
[5] K. Xue *et al.*, "DPSAF: Forward Prediction Based Dynamic Packet Scheduling and Adjusting With Feedback for Multipath TCP in Lossy Heterogeneous Networks," *IEEE Trans. Vehic. Tech.*, vol. 67, no. 2, 2018, pp. 1521–34.
[6] C. Xu, J. Zhao, and G.-M. Muntean, "Congestion Control Design for Multipath Transport Protocols: A Survey," *IEEE Commun. Surveys & Tutorials*, vol. 18, no. 4, 2016, pp. 2948–69.
[7] V.-H. Tran *et al.*, "Observing Real Multipath TCP Traffic," *Computer Commun.*, vol. 94, 2016, pp. 114–22.
[8] A. Nikravesh *et al.*, "An Indepth Understanding of Multipath TCP on Mobile Devices: Measurement and System Design," *Proc. 22nd Annual Int'l. Conf. Mobile Computing and Networking*. ACM, 2016, pp. 189–201.
[9] X. Wang *et al.*, "A Tensor-Based Big-Data-Driven Routing Recommendation Approach for Heterogeneous Networks," *IEEE Network*, vol. 33, no. 1, Jan./Feb. 2018, pp. 64–69.
[10] N. Cheng *et al.*, "Big Data Driven Vehicular Networks," *IEEE Network*, 2018, pp. 1–8.
[11] M. Li *et al.*, "Multipath Transmission for the Internet: A Survey," *IEEE Commun. Surveys & Tutorials*, vol. 18, no. 4, 2016, pp. 2887–2925.
[12] 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, Mar./Apr. 2019, pp. 195–201.
[13] Linux MPTCP kernel; http://www.multipathtcp. org/, accessed Nov. 2019.
[14] L. Guo and I. Matta, "The War Between Mice and Elephants," *Proc. 9th Int'l. Conf. Network Protocols*, IEEE, 2001, pp. 180–88.
[15] M. Allman *et al.*, "TCP Congestion Control," IETF RFC 5681, 2009.

Biographies

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 a graduate student in communication and information systems in the Department of Electronic Engineering and Information Science (EEIS), USTC. His research interests include future Internet architecture and transmission optimization.

In order to provide flexible transmission strategies for various flows in big data, we proposed a cross-layer solution, called FSA-MPTCP. FSA-MPTCP leverages the extra information from applications to customize transmission strategies for different data flows. Our proposal provides better performance for data of different sizes.

Jiangping Han (jphang@mail.ustc.edu.cn) received her B.S. degree from the Department ofEEIS, USTC in 2016. She is currently working toward a Ph.D degree in the same department. Her research interests include future Internet architecture and transmission optimization.

Kaiping Xue [M'09, SM'15] (kpxue@ustc.edu.cn) received his B.S. degree from the Department of Information Security, USTC in 2003 and received his Ph.D. degree from the Department of EEIS, USTC,in 2007. From May 2012 to May 2013, he was a postdoctoral researcher with the Department of Electrical and Computer Engineering, University of Florida. Currently, he is an associate professor in the Department of Information Security and the Department of EEIS, USTC. His research interests include next-generation Internet, distributed networks, and network security. He serves on the Editorial Boards of several journals, including *IEEE Transactions on Wireless Communications*, *IEEE Transactions on Network and Service Management*, and *Ad Hoc Networks* (Elsevier). He has also served as a Guest Editor of the *IEEE Journal on Selected Areas in Communications* and as a lead Guest Editor of *IEEE Communications Magazine*. He is an IET Fellow. He is the corresponding author of this article.

Jianqing Liu (jianqing.liu@uah.edu) received his B.Eng. degree from the University of Electronic Science and Technology of China, Chengdu, in 2013 and received his Ph.D. degree from the Department of Electrical and Computer Engineering, University of Florida, Gainesville, 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.

Miao Pan [S'07, M'12] (mpan2@uh.edu) received his B.Sc. degree in electrical engineering from Dalian University of Technology, China, in 2004, his M.A.Sc. degree in electrical and computer engineering from Beijing University of Posts and Telecommunications, China, in 2007, and his Ph.D. degree in electrical and computer engineering from the University of Florida in 2012. He is now an associate professor in the Department of Electrical and Computer Engineering at the University of Houston. His research interests include cybersecurity, deep learning privacy, big data privacy, cyber-physical systems, and cognitive radio networks. He was an Associate Editor of the *IEEE Internet of Things Journal* from 2015 to 2018.

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