SSMP: Server Selection for Multipath TCP in CDN Environments

Jiayu Yang[1], Jiangping Han[2], Yitao Xing[1], Yuan Zhang[2], Wenjia Wei[2], Kaiping Xue[1,2,*]
[1] School of Cyber Security, University of Science and Technology of China, Hefei, Anhui 230027 China
[2] Department of EEIS, University of Science and Technology of China, Hefei, Anhui 230027 China
[*] kpxue@ustc.edu.cn

GLOBECOM 2020 - 2020 IEEE Global Communications Conference | 978-1-7281-8298-8/20/$31.00 ©2020 IEEE | DOI: 10.1109/GLOBECOM42002.2020.9348043

***Abstract*—Nowadays, mobile devices are equipped with multiple interfaces connected to various networks, which makes it possible to aggregate bandwidth in actual application. Multipath TCP (MPTCP) is one of the transport protocols that uses multiple interfaces simultaneously and provides robust and efficient data transmission. In practice, MPTCP will interact with various network facilities. Among them, Content Delivery Network (CDN) is a popular one, which is a widely distributed network system deployed across the Internet. Using MPTCP in CDN could provide better performance for users, however, we find that CDN may not give full play to its functions when working with MPTCP. Because the Default Server Selection (DSS) mechanism in CDN only obtains servers optimal in single path connection scenarios, it may not provide the globally optimal server for MPTCP. In this paper, we propose a new algorithm called Server Selection for MPTCP (SSMP), which utilizes all available multi-homed sources to provide the globally optimal performance. SSMP modifies the DNS mechanism to return the optimal server for each available interface by the origin strategy and further selects the globally optimal server for both elephant and mice flows. We compare SSMP with DSS through experiments under video streaming and file download scenarios with both stable and variable environments. Our results show that SSMP consistently utilizes available paths more efficiently than DSS, particularly for servers with a great gap in server quality.**

***Index Terms*—MPTCP, CDN, Server Selection**

I. Introduction

Supported by technological progress, mobile devices are equipped with multiple interfaces and enabled simultaneous access to different wireless networks. Multipath TCP (MPTCP) is a standard protocol recommended by IETF to offer parallel data transfer over multiple networks [1], which can not only provide robust transmission but also improve transmission efficiency by aggregating bandwidth of multiple paths. As a practical protocol, MPTCP may have interaction with various network facilities, and Content Delivery Network (CDN) is a popular one, which is an integrated content sharing system deployed across the Internet [2]. Composed with many geographically distributed proxy servers, CDN replicates popular contents at different sites and utilizes a DNS resolution mechanism to resolve Uniform Resource Locator to IP addresses. For each content request, CDN utilizes its Default Server Selection (DSS) mechanism to initiates a DNS request through the default path and prioritizes an optimal server based on various considerations. And content transmission over CDNs has a lower delay and higher stability compared with traditional networks.

Certainly, CDN is compatible with MPTCP, because MPTCP acts the same as single-path TCP from a high-level perspective. But CDN may not take full advantage of its functions when working with MPTCP. As its server selection strategy can only initiate a DNS request through the default path, CDN may only get a sub-optimal server when MPTCP is used in transmission. And here are some specific tests to support our assumption. Nikravesh *et al.* [3] tested the top 500 websites of Alexa in the United States, using WiFi and cellular network to initiate DNS requests respectively to visit the same website, and found that 38% returned different IP addresses. Adarsh *et al.* [4] conducted a survey to explore latency and reach-ability for each of multiple interfaces to the Tranco top 10K websites. They found that there is 45% of the IP addresses of WiFi and cellular networks were different in reachable websites. Therefore, without taking all the available paths under MPTCP into account, the server attained is optimal in single path connection scenarios rather than globally optimal and the server selection strategy in existing CDNs is barely possible to achieve expected performances of MPTCP.

In this paper, we propose a new algorithm Server Selection for MPTCP (SSMP) to choose the globally optimal servers for MPTCP connections. Firstly, SSMP obtains the single-port optimal server for all the available interfaces of MPTCP and detects the path conditions of each server by modifying the DNS sending mechanism. Then considering different link states make disparate effects on the transmission of elephant and mice flows, SSMP gives different strategies for each of them respectively to make further optimization using the detected parameters of each available server. By demonstrating our scheme on video streaming and file download scenarios with both stable and variable environments, we prove that our method achieves significant and consistent improvement compared to the baseline. The main contributions of our work can be summarized as follows:

- We optimize the transmission efficiency of MPTCP in CDN in terms of a new server selection algorithm SSMP.
- By modifying the socket function, we provide a way of initiating multiple DNS requests through available interfaces simultaneously and make it feasible for SSMP to detect path conditions of each available path to CDN servers.
- According to the detected path conditions, SSMP distinguishes flow types and makes specific optimization for server selection. It selects the globally optimal server and provides higher throughput and lower latency for elephant and mice flows, respectively.

The rest of the paper is organized as follows: Section II introduces the related works of MPTCP and CDN in detail,

978-1-7281-8298-8/20/$31.00 ©2020 IEEE

and Section III illustrates the sub-optimal parts of MPTCP transmission with the existing CDN system. We show the improvement approaches for MPTCP transmission in Section IV. Section V shows simulation environments and experiment results, and Section VI serves as a conclusion.

II. Related Work

The demand for high-quality transmission of online content grows rapidly along with the development of the Internet nowadays. And CDN serves a bigger portion of Internet traffic compared with ten years ago when the leading CDN service provider Akamai claimed to share more than 20% of Internet Web traffic [5]. In order to attain higher throughput and guarantee user services, there have been many studies on multipath mechanisms applied to CDN. Chen *et al.* [6] designed an architecture called MSPlayer and used video ID to find the same video for two connections to download simultaneously. But it relies on two TCP transmissions instead of using MPTCP, which will cause unfairness to single TCP flows. Bouttier *et al.* [7] proposed a hybrid architecture allowing efficient routing decisions from both throughput and QoS in a heterogeneous environment. Hayes and Chang [8] proposed a method that addresses long-tail content in CDNs through the implementation of a multipath-aware peer to peer mechanism to distribute a video-specific lightweight neural network model. It actively manages multipath transport networks, but it focuses on the selection of multiple CDNs. All of the three papers use DSS to return servers for user requests and none of them consider the differences of initiating DNS requests through disparate interfaces.

There are no proxy servers in traditional networks, causing the problem of inaccessibility when the only server crashes down for overload requests or breakdowns of the system. In the mechanism of CDN, the same requests can be routed to different servers, which can not only compensate for the server crashed but also provide users with optimal servers. With several proxy servers distributed in multiple geographical locations, CDN utilizes a server selection algorithm to return each user with content requests for a suitable server and the selection of servers has a great impact on the Quality of Service (QoS). So far there have been many algorithms utilizing a DNS resolution strategy to return a proper server in CDN, and they focus on different aspects. For instance, using load information of proxy servers to choose the least loading one in [9], considering Round Trip Time (RTT) between clients and servers to select the server with minimum RTT in [10], and utilizing Round-Robin mechanisms based on a classic Round Robin algorithm or estimating QoS and return the greatest quality one during data transmission in [11] and [12] respectively. However, the transparent design of multiple subflows in MPTCP hides some information from the upper layer. So, no matter which of the strategy is used, the redirection request can only be sent through a default path when we use MPTCP for transmission. Thus, resulting in the inability to obtain global path information and obtaining servers that are sub-optimal. So we provide a new server selection for MPTCP. Compared with the server selection methods mentioned above, the greatest advantage of our algorithm is that not only can it choose these algorithms according to the given environments, but also optimize them further.

III. Problem Analysis of MPTCP with DSS

MPTCP is a mature multipath protocol, which is originated for connection recovery when the links are in a bad condition. It can also provide efficient transmission with the advantage of aggregating bandwidth. To solve the existing problems of traditional multipath transmission protocols and reduce deployment costs, the design that multiple paths conceal beneath an inter-layer attracts much attention. Specifically, MPTCP is divided into MPTCP layer and TCP sub-layer, the former is transparent to the upper application and manages the latter, which is responsible for controlling the established TCP flows on the available paths. Therefore, the application lay does not know the existence of subsequent flows.

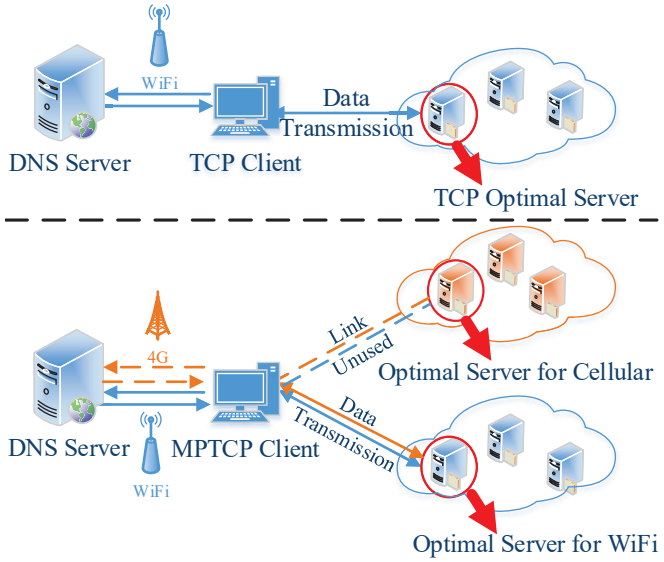

Fig. 1. Description of suboptimal problem.

As shown in Fig. 1, CDN utilizes its server selection mechanism and returns a reasonable server for every DNS request by considering content storage, link states and server states, etc. Usually, we think that the server selected is the best one for the interface at that time. But is it optimal when using MPTCP to transfer data without changing the server selection strategy of CDN? The work in [13] found there are many differences between WiFi and cellular networks. And Li *et.al.* [14] illustrated that the networks offered by different network carriers present immensely disparate characteristics. It is indicated that the server returned for WiFi is barely the optimal one for cellular transmission, which means the server attained by DSS is only optimal in single path connection scenarios rather than globally optimal. In addition to the works mentioned above, there are specific tests to prove our assumption. Nikravesh *et al.* [3] used WiFi and cellular network to initiate DNS requests respectively to visit the same

website, and 38% among the top 500 websites of Alexa in the United States returned different IP addresses. Besides, Adarsh *et al.* [4] conducted a survey to explore the states of latency and reach-ability of multiple interfaces to the Tranco top 10K websites. They found that after DNS resolution, the reachable websites for Wi-Fi and LTE have a difference of 41.26% in IP addresses.

Considering that even if different IP addresses are returned, they may be different virtual machines on the same host or different servers in the same geographical location. We used an app called best trace, which can not only show DNS resolution results but also present server locations, to further prove our assumption. Huawei network card e8372-855 and a mobile card are inserted into Lenovo Y900X to access the mobile 4G signal, and the telecom WiFi signal is connected to the telecom router. Then we initiated DNS requests to the top 10 websites of Alexa in China, using WiFi and 4G connection respectively. We found that nine of them return different IP and four of them in different provinces. This strongly proves that the servers obtained by using the original server selection system of CDN are sub-optimal for MPTCP. So we need a new server selection strategy to get the globally optimal server and improve transmission efficiency for MPTCP in CDN environments.

IV. Our Solution

Origin server selection schemes can only send DNS requests through a default path and get servers optimal in single-path connection scenarios. In this part, we introduce our scheme SSMP, which can leverage multiple interface resources to select the globally optimal servers for MPTCP in CDN environments. Firstly, we propose a method of initiating multiple DNS requests to obtain the sub-optimal server of all available interfaces. And then we present server selection strategies to further select the globally optimal servers for both elephant and mice flows specifically.

A. Multiple DNS Requests

Firstly, we introduce a scheme of initiating multiple DNS requests for the same content. In the current DNS mechanism, DNS requests are usually transmitted through UDP and one DNS response matches one DNS request. To finish multiple server request processes within the time of one DNS resolution, each interface with different networks is controlled to initiate DNS requests simultaneously. So we can get multiple suboptimal servers and make it feasible for SSMP to send packets to probe corresponding path information of each available path as soon as we received the applied IP addresses. With additional recorded path information, we can choose the globally optimal server according to our server selection strategies, which will be presented in the next subsection.

B. Server Selection Strategies

We propose different server selection strategies for mice and elephant flows because RTT and bandwidth have disparate impacts on the transmission rate for them. Specifically, the transmission rate of mice flows is mainly affected by RTT, while that of elephant flows is mainly affected by bandwidth. Besides, CDN can deliver the content size in content reply packets, so we can determine the flow type and choose the corresponding strategy.

1) Strategy for Elephant Flows: In the way of comparing the IP addresses returned, we can determine whether the sub-optimal servers of available interfaces are the same or not. If so, we use a random selection strategy. If not so, in order to select servers with a stronger ability of aggregating bandwidth, we define the QoS for MPTCP servers in consideration of the following principle of transmission.

Under the condition of unlimited bandwidth, the maximum throughput of a TCP connection can be estimated as follow [15],

$$T = \frac{MSS * C}{RTT * \sqrt{P}}, \tag{1}$$

where P denotes packet loss rate of the link that can be calculated from historical data collected for preparation and RTT can be obtained by sending ping packets. The value of MSS is 1460 bytes in our environment and parameter C is a constant whose value is $\sqrt{3/2}$ if the link suffers random packet los. The specific information about C can be obtained in [15]. As for elephant flows, we mainly concern the impact of bandwidth. So we consider the throughput constraint and add the subflow heterogeneous factor which affects the performance of MPTCP connections to define the QoS of servers, which presents as follows:

$$Q = \sum_{i=1}^{n} \beta_i * min(T_i, B_i), \tag{2}$$

where n is the number of subflows of MPTCP connection, and i is the sequence of subflows. B denotes the average limitation of the achievable bandwidth based on the previous transmission, which is the upper bound when there is no packet loss. And $\beta_i = min(RTT_1, \cdots, RTT_n)/RTT_i$ is the heterogeneous factor between subflows. And T denotes the estimated throughput of a TCP connection obtained by the information of links, whose design formulation is shown in Equation (1). To implement the algorithm, DSS is used in the beginning when there is no effective previous estimation of link information, and we start to use our strategy SSMP until the initial value of B and P are obtained.

The strategy above is committed to a stable network environment. However, the link state is changing constantly in real Networks, and there are deviations in testing data, which need to be updated. Therefore, according to the trustworthiness of the path state, a new selection scheme is proposed. QoS for servers using MPTCP, which is modified according to new requirements and devoted to the variable environment is

$$Q_n = \alpha Q_{new} + (1 - \alpha) Q_{n-1}. \tag{3}$$

The calculation method of Q_{new} is the same as Q in a stable situation, representing the current service quality of servers. And Q_{n-1} denotes the last QoS from historical data. Among

them, α denotes the worthiness of the current estimated value Q_{new}. When the network changes smoothly, we propose a rather radical method by giving a larger weight to Q_{new}. On the contrary, a smaller α is recommended for the network whose status change obviously. And we update the values of B and Q_{n-1} whenever the transmission is completed to renew the state of servers timely. However, this policy is useless for servers with rather old historical data. Thus, we send probe packets to update the data When a server is not been used for a long time. This strategy may have some disadvantages for the first time to access to the server that has not been enabled for a long time, but it can effectively prevent servers from starving and guarantee the best service for the following transmission.

2) Strategy for Mice Flows: In general, the transmission of mice flows will not reach the limit of bandwidth, so our strategy is to ensure flows transfer on paths with a shorter RTT. And the definition of QoS of servers is $q = \sum_{i=1}^{n}\{RTT_i\}/n$, where n denotes the number of available subflows of a sub-optional server. And RTT_i denotes the latest RTTs of $subflow_i$, which can be obtained by sending ping packages within two or three RTTs. On the contrary to elephant flows, we choose the server with a smaller q value as the optimal server. Besides, we turn to select the server with the shortest RTT of all available paths if the q values are the same. However, according to the implementation codes of MPTCP in the Linux kernel, the initial subflow can transmit a part of the data previously compared with subsequent subflows, which has a significant influence on the transmission of mice flows. Therefore, considering the establishing order of the subflows will also affect the transmission efficiency, we establish the initial subflow of MPTCP using the path with the smallest RTT in the available paths of the selected server to make further optimization.

V. Performance Evaluation

We evaluate SSMP under various scenarios including different network conditions (stable *and* dynamic) and different workloads (video streaming *and* file download). Our testbed includes six PCs running on 64 bits Ubuntu 16.04 with the MPTCP 0.94 [16] implementation, and their functions are as follows, one client, two routers and three servers using Apache2. It should be noted that, we only use three servers to simulate a CDN environment limited to resources. But SSMP selects the globally optimal server out of available sub-optimal servers, so this simple environment is fully to demonstrate the effect of the algorithm. Besides, all of the three servers have the same scheduling method and congestion control algorithm to ensure the impact of our strategy. Paths are emulated by a network emulator, which uses Linux tc to throttle the bandwidth and add extra delay or packet loss on the client-server paths. Moreover, we use the random server selection strategy to return the single-port optimal server, and we provide both WiFi and Cellular for the three sub-optional servers. The topology of the experiment is shown in Fig. 2.

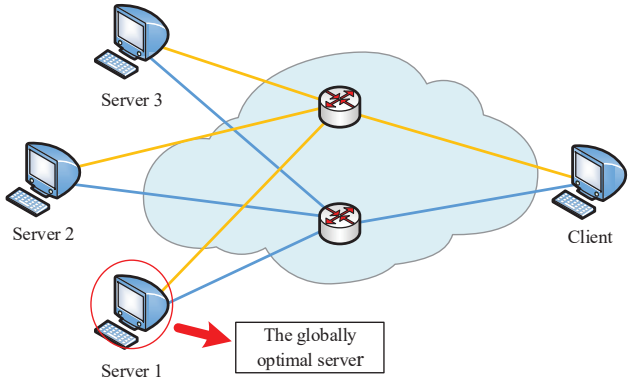

Fig. 2. Topological graph of simulation environment.

A. Video Streaming

Firstly, we show the efficiency of our strategy on video streaming. The video player we used is MP4Client and we get the playback latency from log records. We find that every video presents periodic downloading or on-off cycles during playing, specifically, the video player pauses chunk retrieval when the playout buffer is full and resumes chunk retrieval when the amount of buffered video falls below a certain level. As the buffer size is less than 1 MB, so we utilize the strategy for mice flow to chose the globally optimal server. On the link side, we consider different average latency with the same aggregate bandwidth and the counter. In the following, we show link information of two paths of each server Specifically. *Server1* and *server3* have identical bandwidth values of 40 Mbps and 20 Mbps, but different RTT values of 5 ms and 15 ms compared with 30 ms and 30 ms. *server2* is a comparison with bandwidth values of 40 Mbps and 30 Mbps and delay values of 10 ms and 10 ms. Before showing the efficiency of the algorithm, we present the performance of three servers when streaming two videos of different sizes, shown as Fig. 3.

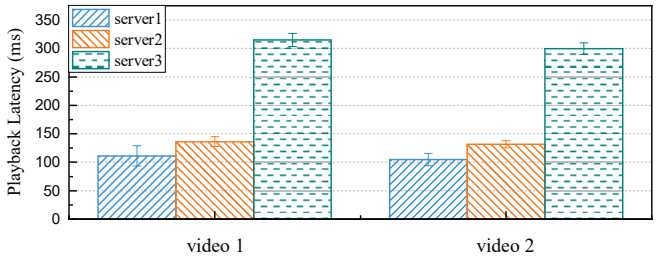

Fig. 3. Performance of servers transfering different sizes of videos.

From Fig. 3 we can see that video size and bandwidth make little influence on playback latency, but latency makes the decisive influence, which is consistent with our strategy. And then we show the efficiency of our strategy using DSS and SSMP to stream video1, and both of them repeat the streaming process for fifty times, and the CDF of the playback latency is shown as Fig. 4.

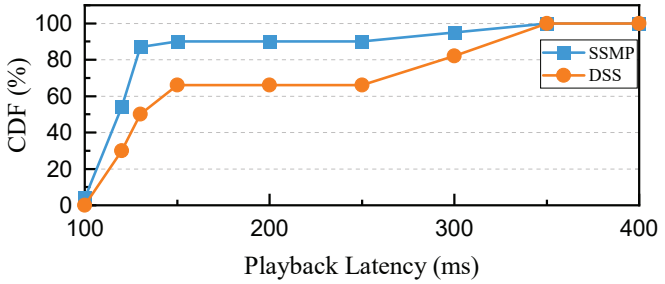

Fig. 4. Performance of SSMP in video streaming.

From Fig. 4, we can see, More than 80% of the playback latency of SSMP is less than 130ms, while that of DSS is only 50%. And 90% of the playback latency of SSMP is less than 150ms, while that of DSS is only 66%. The result reveals that our strategy makes full use of the most optimal server and provides better efficiency.

B. File Download

In this part, we show our evaluation focuses on file download, and we present the efficiency of the algorithm by comparing the download time of the same files using different strategies. The baseline uses DSS and transmits data by MPTCP. The topology is the same as above. In order to ensure the accuracy of data, we download each file for ten times and report maximum, minimum and average value.

1) Stable Environment: We begin by investigating whether SSMP improves the performance of file delivery compared to DSS in a stable environment using emulation. We consider the impact of delay, bandwidth and packet loss of the paths.

Mice Flows: We first test the efficiency of the strategy for mice flows using the same link information as video streaming. The corresponding results are shown in Fig. 5. We

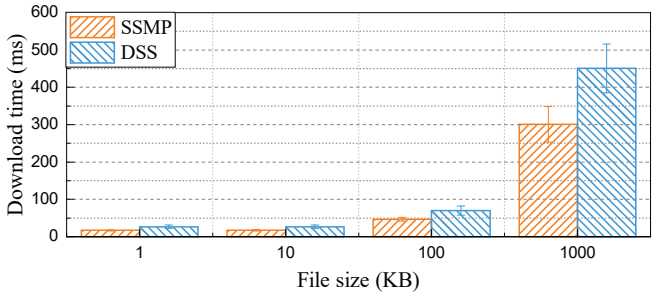

Fig. 5. Efficiency of strategy for mice flows.

can see that SSMP outperforms DSS in terms of download time. Specifically, under such circumstances, SSMP reduces download time by 38.46% for 1 KB and 10 KB, and reduces by 34.28% and 33.11% for 100 KB and 1000 KB respectively. It observes that the shortest RTT is decisive for flows less than 10 KB because MPTCP completes transmission before using the subsequence subflows. We can also draw that when servers have the same aggregate bandwidth and different latency, the file download rate is different, which further proves that aggregate bandwidth has little effect on download time for mice flows. And SSMP brings more reduction in download time as the differences of average RTTs between servers become larger.

Elephant Flows: Compared with mice flows, we add loss rate into consideration. As for link states, we consider the effects of different aggregation bandwidth and the influence of packet loss. Specifically, *Server1* and *Server2* have identical RTT values of 10 ms and 20 ms, but different bandwidth values of 80 Mbps and 30 Mbps compared with 40 Mbps and 20 Mbps. *Server3* is a contrast whose bandwidth and delay values are as same as *Server2*. And in order to analyze the influence of packet loss, *Server3* suffers an extra 1% and 2% loss rate for each path compared with *Server1* and *Server2*. The relevant results are shown in below. From Fig. 6 we can

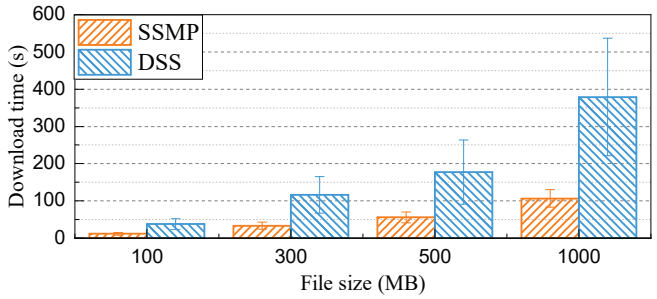

Fig. 6. Efficiency of strategy for elephant flows in stable network.

see that, under such circumstances, SSMP is superior to DSS in file transmission. SSMP reduces download time by about 65% for 100 MB and 300 MB, and reduces by 70% and 66% for 500 MB and 1000 MB respectively. Compare with Fig. 5, elephant flows have a better performance in reducing download time, but suffer a great gap in the max and min values. This is because *Server3* suffers packet loss, which leads to the decline of service quality. So it has a great gap compared with the first two servers, but DSS is unable to avoid the worst server causing great fluctuation. While proving the efficiency of the algorithm, we also verify that when the bandwidth is the main factor, delay and packet loss rate have little impact on the transmission rate. However, when delay and packet loss rate become the main factors, the aggregate bandwidth has little effect on the transmission rate. Our results further show that aggregate bandwidth plays a decisive role in the transmission rate of elephant flows only under certain conditions.

2) Dynamic Network Situations: We next show the efficiency of SSMP under changing network conditions. For *Server1*, we simulate the case that the available bandwidth reduces and the value of RTT increases suddenly due to overloading. We experiment based on the previous test and the states of *Server2* and *Server3* are unchanged. For each path of *Server1*, packet loss rate remains unchanged, while latency increases 10 ms. And the values of bandwidth reduce to 20 Mbps and 30 Mbps, which means that its QoS is worse than *Server2*. We take downloading 300 MB and 500 MB files as examples to show the impact of α on transmission. From

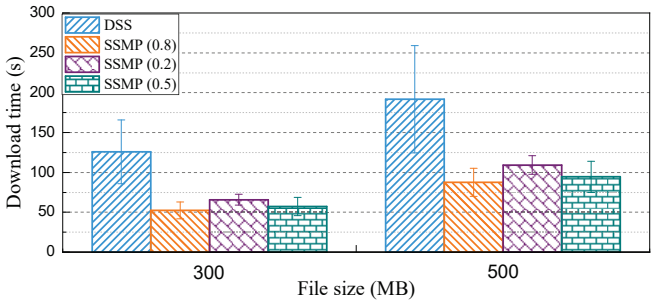

Fig. 7. Efficiency of strategy in changing environment.

Fig. 7, we can see that SSMP also has a significant effect in reducing download time as a stable environment. When α is 0.8, SSMP performs best, followed by 0.5 and 0.2 is the worst. The reason for this difference is that when the QoS of *Server1* becomes poor, it will take some time to complete the transformation of the optimal server, and 0.8 can complete the switch more timely than 0.2. But it's not that the faster the handoff the better performance, because there is a deviation of the tested data in dynamic networks. In the following part, we discuss the situation that one link of *Server1* tested 1% packet loss rate wrongly, and the result is shown as Fig. 8. We can see that it presents a completely different situation.

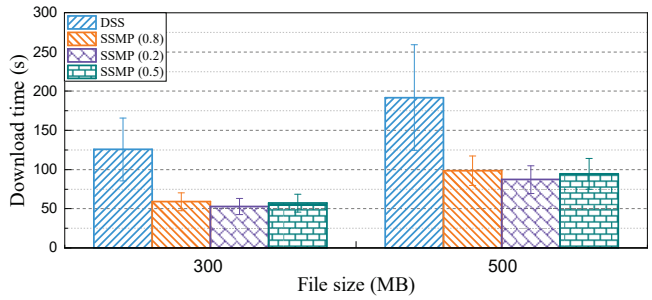

Fig. 8. Efficiency of strategy in testing error environment.

When α is 0.8, SSMP performs the worst and 0.2 is the best. Therefore, the value of α can be adjusted according to the actual situation. Although the different values of α have some influence on the transmission, our strategy is far better than DSS even when α is in the worst situation.

The test results in the changing environment reveal that our strategy can not only avoid the worst link timely but also guarantee the best service.

VI. Conclusions

In this paper, we proposed a way of initiating multiple DNS requests and obtaining multiple sub-optimal servers within the time of a normal DNS request process. And we also provided a new server selection strategy SSMP for mice and elephant flows to select the globally optimal servers by considering all the available multi-homed sources. According to the experimental results, the optimization schemes proposed by us have a significant effect on shorting video playback latency and file download time no matter in a stable or dynamic environment. Compared with CDN using TCP protocol and DSS, using MPTCP in CDN has a qualitative improvement in the efficiency of content delivery. And our strategy can further improve the transmission efficiency for MPTCP in CDN environment.

Acknowledgment

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

References

[1] D. Wischik, C. Raiciu, A. Greenhalgh, and M. Handley, "Design, implementation and evaluation of congestion control for Multipath TCP." in *NSDI*, vol. 11, 2011, pp. 8–8.

[2] A. O. Al-Abbasi, V. Aggarwal, and M. Ra, "Multi-tier caching analysis in cdn-based over-the-top video streaming systems," *IEEE/ACM Transactions on Networking*, vol. 27, no. 2, pp. 835–847, 2019.

[3] A. Nikravesh, Y. Guo, F. Qian, Z. M. Mao, and S. Sen, "An in-depth understanding of multipath TCP on mobile devices: Measurement and system design," in *Proceedings of the 22nd Annual International Conference on Mobile Computing and Networking (Mobicom)*. ACM, 2016, pp. 189–201.

[4] V. Adarsh, P. Schmitt, and E. Belding, "MPTCP performance over heterogenous subpaths," in *2019 28th International Conference on Computer Communication and Networks (ICCCN)*, 2019, pp. 1–9.

[5] E. Nygren, R. K. Sitaraman, and J. Sun, "The Akamai network: a platform for high-performance internet applications," *Acm Sigops Operating Systems Review*, vol. 44, pp. 2–19, 2010.

[6] Y.-C. Chen, D. Towsley, and R. Khalili, "MSPlayer: Multi-source and multi-path video streaming," *IEEE Journal on Selected Areas in Communications*, vol. 34, no. 8, pp. 2198–2206, 2016.

[7] E. Bouttier, R. Dhaou, F. Arnal, C. Baudoin, E. Dubois, and A. L. Beylot, "Analysis of content size based routing schemes in Hybrid Satellite / Terrestrial Networks," in *Proceedings of 2017 IEEE Global Communications Conference (Globecom)*, 2017.

[8] B. Hayes and Y. Chang, "Lightweight evolving 360 VR adaptive video delivery," in *2020 International Conference on Computing, Networking and Communications (ICNC)*, 2020, pp. 815–819.

[9] X. Zhou, T. Dreibholz, and E. P. Rathgeb, "A new server selection strategy for reliable server pooling in widely distributed environments," in *2nd Proceedings of International Conference on the Digital Society (ICDS)*. IEEE, 2008, pp. 171–177.

[10] Y.-T. Han, M.-G. Kim, and H.-S. Park, "A novel server selection method to achieve delay-based fairness in the server palm," *IEEE Communications Letters*, vol. 13, no. 11, pp. 868–870, 2009.

[11] P. Wendell, J. W. Jiang, M. J. Freedman, and J. Rexford, "DONAR: decentralized server selection for cloud services," *Proceedings of the 2010 ACM Special Interest Group on Data Communication (Sigcomm)*, vol. 41, no. 4, pp. 231–242, 2011.

[12] H. A. Tran, S. Hoceini, A. Mellouk, J. Perez, and S. Zeadally, "QoE-based server selection for content distribution networks," *IEEE Transactions on Computers*, vol. 63, no. 11, pp. 2803–2815, 2013.

[13] Y.-C. Chen, Y.-s. Lim, R. J. Gibbens, E. M. Nahum, R. Khalili, and D. Towsley, "A measurement-based study of multipath TCP performance over wireless networks," in *Proceedings of the 2013 conference on Internet measurement conference (IMC)*. ACM, 2013, pp. 455–468.

[14] L. Li, K. Xu, T. Li, K. Zheng, C. Peng, D. Wang, X. Wang, M. Shen, and R. Mijumbi, "A measurement study on multi-path TCP with multiple cellular carriers on high speed rails," in *Proceedings of the 2018 Conference of the ACM Special Interest Group on Data Communication (Sigcomm)*. ACM, 2018, pp. 161–175.

[15] N. Kai, H. Yoshida, and K. Nihei, "An ensemble method for estimating TCP throughput on application layer," in *15th IEEE Annual Consumer Communications Networking Conference (CCNC)*, 2018, pp. 1–4.

[16] Linux MPTCP kernel. Accessed: Jan. 2020. [Online]. Available: http://www.multipattcp.org/.