Performance Evaluation of MPTCP over a Shared Bottleneck Link

Soonghwan Ro*, Dien Nguyen Van
Department of Information and Telecommunication, Kongju National University, No.275 Budae-dong, Cheonan, Chungnam, 330-717, Korea.

* Corresponding author. Email: rosh@kongju.ac.kr
Manuscript submitted November 25, 2014; accepted June 8, 2015.
doi: 10.17706/ijcce.2016.5.3.176-185

Abstract: In this paper, we present experimental results evaluating the performance of the Multipath TCP over a shared bottleneck path in a series of benchmark tests. We found that the Multipath TCP's fairness as well as its competitiveness responds to the change of network conditions, such as latency and loss rate. MPTCP is unfairness and powerful with regular TCP in ideal network conditions, but its throughput clearly decreases even less than regular TCP in worse network conditions with high latency and a higher rate of loss of packets.

Key words: Performance of multipath TCP, shared bottleneck, fairness.

1. Introduction

The Multipath TCP (MPTCP) [1] protocol has recently been standardized by the Internet Engineering Task Force (IETF) and an implementation in the Linux kernel is now available [2]. MPTCP allows running over several Internet paths between a pair of hosts, while still presenting as a single TCP connection to the application layer. More than 95% of total Internet traffic is still driven by TCP. Hence, primarily MPTCPs are required to operate over any Internet paths where TCP exists. Consequently, question that has arisen is how to examine the MPTCP's performance and behavior with the regular TCP. This implies that evaluation of the MPTCP's performance in various real networks is necessary before it can be deployed extensively.

In order to answer the above question, we conducted some scenarios in which there was competitive traffic between the regular TCP and the MPTCP connections over a shared bottleneck path. Also, we varied the parameters to emulate real-world conditions. The results were evaluated in terms of throughput and fairness.

The rest of this paper is organized as follows. In Section 2, we summarized the previous work relevant to this paper. In Section 3, we introduced background information about MPTCP. In Section 4, the experimental conditions and the analytical results are presented. In Section 5, we discuss possible future work and present our conclusions.

2. Related Works

In recent years, the subject of multi-homing, especially MTCP has received a lot of attention among the research community, and the approach was deployed by an IETF working group. The Stream Control Transmission Protocol (STCP) [3] was designed with consideration for multi-homing and resolving the path failure. SCTP enables the host to use multiple paths at the same time. Although SCTP is implemented in

several operating systems [4], it is still not used extensively except for specific applications. There are two main disadvantages of SCTP on the global Internet. First, the developers of applications must change them before they can use SCTP. Second, various types of middle boxes, such as Network Address Translation NAT or firewalls, do not understand SCTP and block all SCTP packets. Since 2012, the MPTCP working group of the IETF has been developing multipath extensions to TCP that enable the hosts to use several paths, possibly through multiple interfaces, to carry the packets that belong to a single connection. Additionally, these extensions solve the SCTP's drawback of requiring changes in applications and of conflicts with middle boxes. This is probably the most ambitious extension to TCP to be standardized within the IETF. Based on this implementation, some researchers [5] have shown the impacts of various parameters on MPTCP's performance. This information is extremely useful for understanding MPTCP and conducting our experiments.

Although there are many issues that limit laboratory measurements, the results we achieved will be of significant assistance to the networking community in estimating the MPTCP's performance, which is a new protocol at the transport layer.

3. MPTCP: Overview

3.1. Architecture

MPTCP is a new transport protocol designed to transport signaling traffic. The significant motivation was to allow a multi-homed host to spread a single TCP connection across multiple interfaces. This will help MPTCP achieve significant benefits, e.g., maintaining a connection when links fail; using congestion control to steer traffic away from congested links, thereby reducing congestion; and increasing efficiency by taking advantage of additional interfaces with parallel paths. The design of MPTCP has been influenced by many requirements, but there are two requirements that stand out from the others, i.e., application compatibility and network compatibility. Application compatibility implies that an application that runs with TCP can run with MPTCP without any changes. The network compatibility goal requires that MPTCP safely pass through all middle boxes. Consequently, MPTCP is designed as a modification of the exits of TCP with an additional option in case the MPTCP connection seems to be a regular TCP connection to an application. However, to the network layer, each MPTCP sub-flow looks like a regular TCP flow, the segments of which carry a new TCP option. Multipath TCP manages creation, utilization, and removal of these sub-flows to send data. The number of sub-flows that are managed within a Multipath TCP connection is not fixed, and it can fluctuate during the lifetime of the Multipath TCP connection. The MPTCP's stack is shown in Fig. 1.

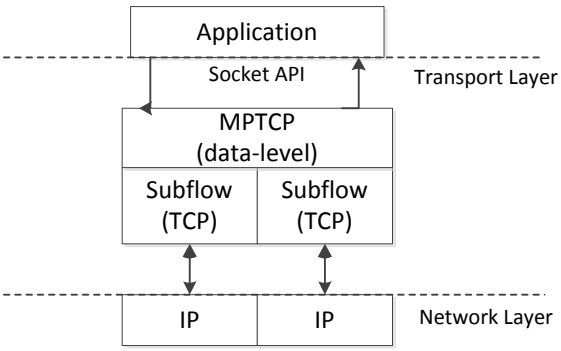

Fig. 1. MPTCP protocol stack.

3.2. MPTCP Operation

Multipath TCP is a set of TCP extensions defined in RFC 6824 that allow a single TCP connection to send

and receive data simultaneously using different IP addresses. It is necessary to present its mechanics, so an outline of its operation is presented in this section. As is the case for a regular TCP connection, an MPTCP connection's lifetime also includes three phases, i.e., initial connection, data transfer, and closing connection. In the initial phase, the difference between MPTCP and TCP is that MPTCP has a four-way handshake before the multipath really is enabled. First, it is similar to TCP with a three-way handshake. But the SYN, SYN/ACK, and ACK packets also carry the MPTCP_CAPABLE option. Second, after the TCP connection has been established, the client can advertise its other address by sending TCP segments with the ADD_ADDR option.

The new address's exchange is used to open a second TCP connection, which is called a new sub-flow, and it will be linked to the first sub-flow by sending a SYN segment with the MP_JOIN option that contains the token sent by the server in the initial sub-flow. The server replies by sending a SYN+ACK packet with the MP_JOIN option that contains the token chosen by the client in the initial sub-flow, and the client terminates the three-way handshake. Fig. 2 describes this process.

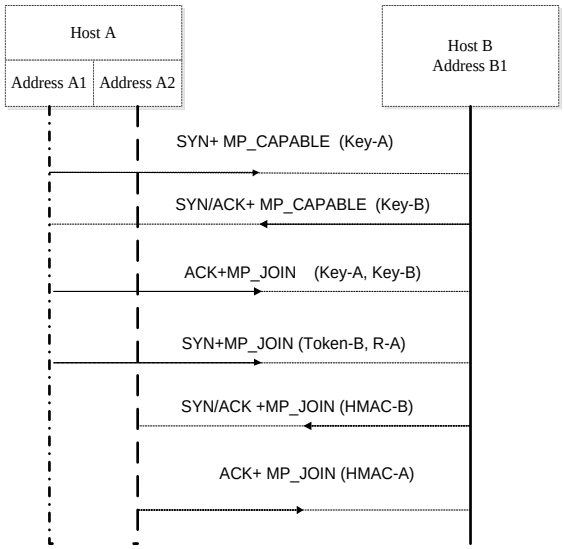

Fig. 2. Initial MPTCP connection.

These two sub-flows are linked together inside a single Multipath TCP connection, and both can be used to send data in the second phase. At the moment, the MPTCP connection enters the data transferring phase. To ensure reliable, in-order delivery of data over sub-flows that may appear and disappear at any time, MPTCP uses two principles. First, each sub-flow is equivalent to a regular TCP connection with its own 32-bit sequence numbering space. This is important to allow Multipath TCP to traverse complex middle boxes like transparent proxies or traffic normalizers. Second, MPTCP maintains a 64-bit data sequence numbering space. Two TCP options use these data sequence numbers, i.e., DSN_MAP and DSN_ACK. When a host sends a TCP segment over each sub-flow, it uses the DSN_MAP option for mapping between the 64-bit and the 32-bit sequence numbers used by the sub-flows. In this way, data can be retransmitted on different sub-flows (mapped to the same DSN) in the event of failure because of reordering the data received, possibility out-of-sequence over different sub-flows. In Multipath TCP, a received segment is acknowledged at two different levels. First, the TCP cumulative or selective acknowledgments are used to acknowledge the reception of the segments on each sub-flow. Second, the DSN_ACK option is returned by the receiving host to provide cumulative acknowledgment at the data-sequence level. When a segment is lost, the receiver detects the gap in the 32-bit sequence number that is received, and traditional TCP retransmission mechanisms are triggered to recover from the loss. When a sub-flow fails, Multipath TCP detects the failure and retransmits the unacknowledged data over another sub-flow that is still active.

When a sender wants to inform the receiver that no more data will be sent, the Data FIN option is signaled as part of the Data Sequence Signal. It has the same semantics and behavior as a regular TCP FIN, but it belongs to the connection level. After all of the data on the MPTCP connection has been received successfully, then this message is acknowledged at the connection level with a DATA_ACK. A connection is considered closed after both hosts' DATA_FINs have been acknowledged by DATA_ACKs. And both hosts should send FINs on all sub-flows as described in Fig. 3. Another important difference between Multipath TCP and regular TCP is the congestion-control scheme. To avoid being unfair with the regular TCP flows, Multipath TCP should not use the standard TCP congestion control scheme, especially when the flows compete with each other on a shared bottleneck path. Linked Increases Algorithms (LIA) was designed to resolve this problem [6], and they rely on congestion control algorithms. Many studies already have been conducted to show the effectiveness of LIA. Therefore, the LIA congestion control scheme was used in this paper.

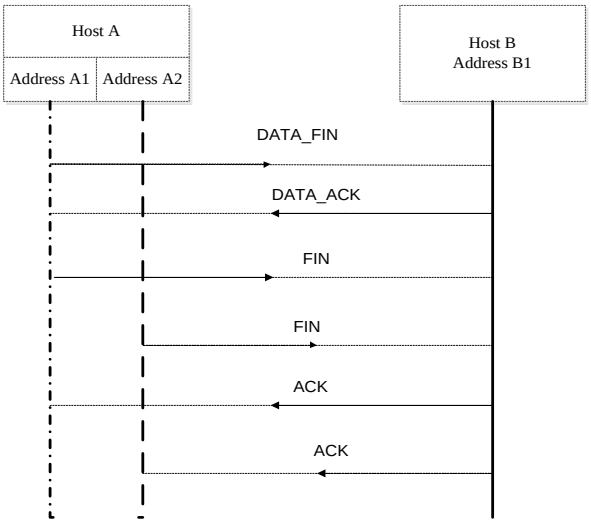

Fig. 3. Closing an MPTCP connection.

Table 1. MPTCP Options

Name	Description
MPTCP_CAPABLE	Multipath capable
MP_JOIN	Join connection
DATA_SEQUENCE_SIGNAL	Data ACK and Data Sequence Mapping
ADD_ADDR	Add Address
REMOVE_ADDR	Remove Address

4. Implementation and Experiments

4.1. Environment in Which the Experiments Were Conducted

This section provides information concerning a series of tests. We conducted to measure the throughput and fairness index of MPTCP. We created a bottleneck shared between the users of MPTCP and TCP. We controlled the network's conditions by imposing latency and increasing packet losses using a dummy-net bridge [7]. Latencies of Internet connections of less than 100 ms are normal, but, usually, it is desirable that they be less than 25 ms. The average latency of a satellite Internet connection is always around 500 ms or greater [8]. Therefore, our experiment was conducted in excess of the above latencies. Each loss was considered as a congestion event in the network. Normally, the rate of packet loss is less than 1%, which is considered to be "good," with rate between 1.0 and 2.5% being considered "acceptable." The simulated rate

of packet loss was selected randomly across four orders of magnitude, spanning roughly in an interval of [0.0003, 0.3].

In order to measure fairness, we used Jain's Fairness Index (JFI) [9] for average user throughput, which is given by:

$$\text{Fairness (throughput)} = \frac{\{\sum_i^n T_i\}^2}{n*\{\sum_i^n T_i^2\}} \quad (1)$$

where T_i is the throughput of the i_{th} connection, and n is the number of competing objects.

Fig. 4 shows the topology of the network.

In this test bed, all links were configured to a bandwidth of 100 Mbps. The bridge, running dummy-net software, can be configured with various round-trip propagation delays and packet loss rates to emulate a wide range of network conditions. Also, an important function is an IP address filter (ipfw), which can affect each sub-flow independently. Both the MPTCP client and server ran the *mptcp-kernel* under Ubuntu 13.10 and used the Linked Increase Algorithm (LIA), which is defined as the standard congestion control scheme of MPTCP. The TCP client used the Reno scheme. The server and the clients used *iperf*, which is a tool for measuring the network's performance. In our research, we conducted two kinds of networks, i.e., a homogeneous network and a heterogeneous network. In the homogeneous network, all sub-flows were similar in latency and packet loss rate. In the heterogeneous network, we used an additional filter of the bridge to create an unbalanced condition between the two sub-flows of the MPTCP connection. In detail, we changed the first sub-flow's latency and loss rate and did not touch the second sub-flow and TCP connection. Therefore, the second sub-flow and TCP connection were always in good condition with less than 1 ms of latency and 0% of packet loss rate, while the first sub-flow varied from good quality to bad quality. Five measurements were made of the same parameters, and each measurement took 10 minutes. Also, we presented the average results.

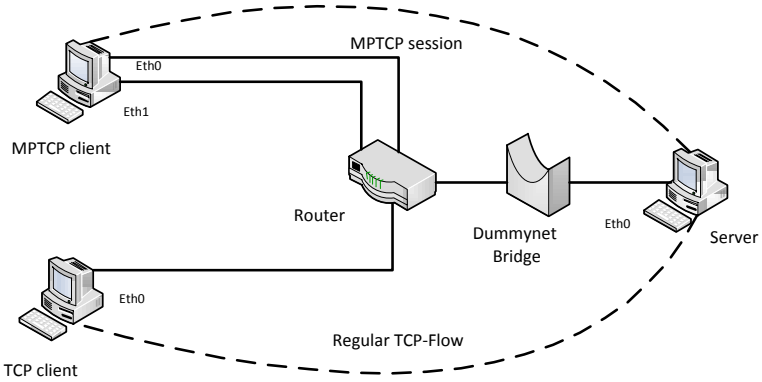

Fig. 4. Implementation scenario.

4.2. MPTCP's Performance in the Homogeneous Network

4.2.1. The homogenous case with same propagation delays

In the first dynamic test, we only changed the network's latency. The TCP and MPTCP clients started and finished at the same time. We obtained the trajectories of the individual connection throughputs (in Mb/s) over time from the sender. They are shown in Fig. 5. In good condition network (latencies less than 100 ms), the throughput of MPTCP was much greater (approximately double) than TCP's throughput. However, as the RTT increased, the bandwidth of MPTCP approached that of TCP. Even so, MPTCP's throughput was a little less than TCP's throughput when the RTT was increased to 200 ms. We calculated Jain's fairness indexes

from the throughput trajectories of each protocol, and they are shown in Fig. 6. It is clear that MPTCP obtained the best fairness in the bed condition, i.e., very close to 1. As shown in Fig. 5 and Fig. 6, the performance of MPTCP decreased as the propagation delay increased.

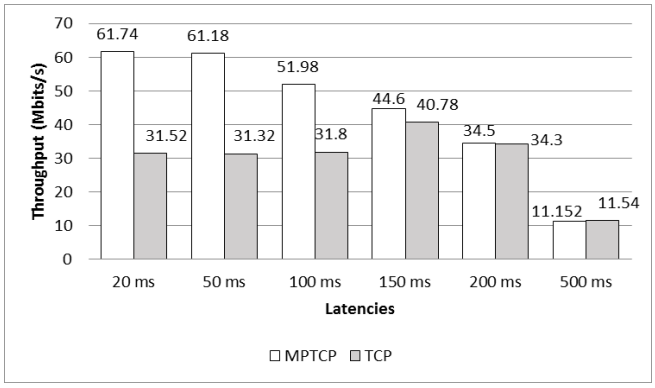

Fig. 5. Comparison of the throughputs of MPTCP and TCP connection.

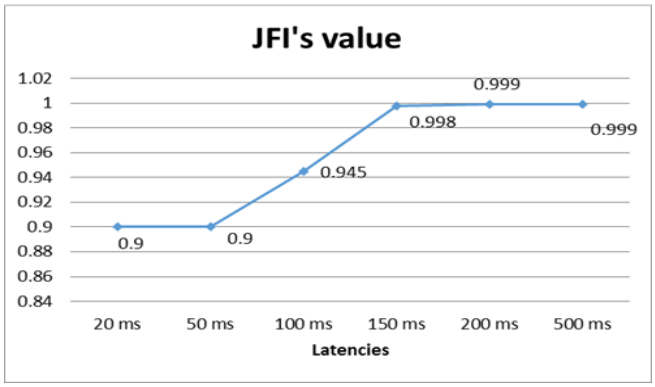

Fig. 6. Jain's fairness indexes.

4.2.2. The homogenous case with same packet loss rates

This experiment was similar to the first dynamic test, but, instead of latency, we only varied the packet loss rate of all of the flows. The throughput obtained and the calculated values of Jain's fairness indexes are shown in Fig. 7 and Fig. 8, respectively. The figures shape look like the results in the first dynamic test. When the packet loss rate is small, the MPTCP's throughput is extremely larger than TCP's throughput. However the unbalance reduces gradually when the loss rate increases. It means that that the better the network conditions are, the more competitive the MPTCP connection becomes.

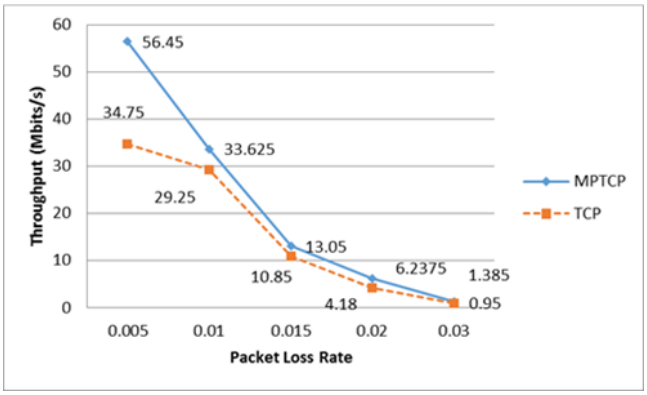

Fig. 7. Comparison of throughput between MPTCP and TCP connection.

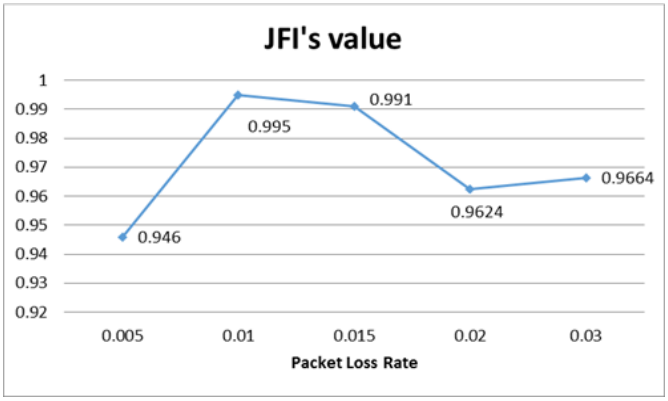

Fig. 8. Value of Jain's fairness indexes.

4.3. MPTCP's Performance in the Heterogeneous Network

4.3.1. The heterogeneous case with different propagation delays

In this experiment, we only changed the first sub-flow's latency and did not change anything to second sub-flow and TCP flow as explained in section A. Fig. 9 shows that the different latency levels between the two sub-flows affect MPTCP's throughput. In detail, when the difference is small (less than 100 ms), the throughput of the MPTCP connection was greater than that of the normal TCP. However, it gradually became more balanced with TCP as the RTT of one sub-flow increased greatly. The MPTCP's throughput decreased as the difference became greater over 200 ms. Jain's fairness indexes shown in Fig. 10 demonstrate that fairness is also small when the difference is small.

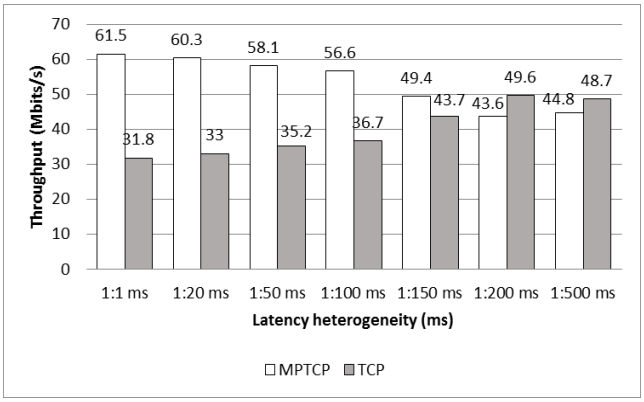

Fig. 9. MPTCP's throughput in terms of unbalanced latency levels of sub-flows.

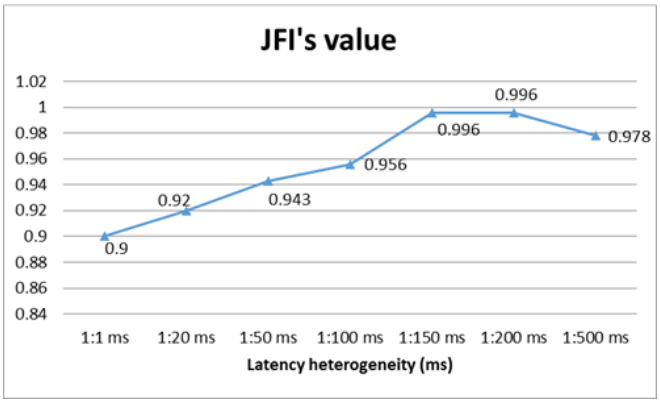

Fig. 10. Jain's fairness indexes in terms of unbalanced latency level of sub-flows.

4.3.2. The heterogeneous case with different packet loss rates

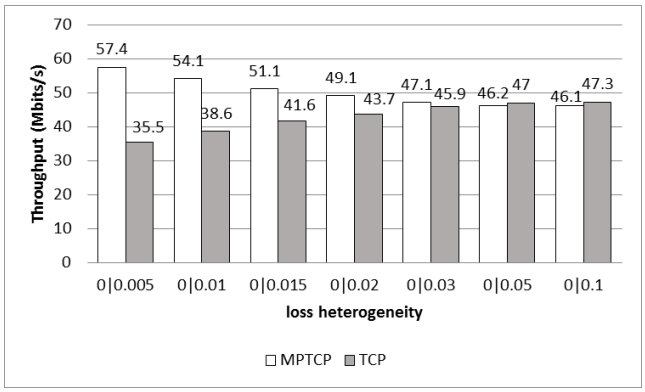

Fig. 11. MPTCP's throughput in terms of unbalanced packet loss rate levels of sub-flows.

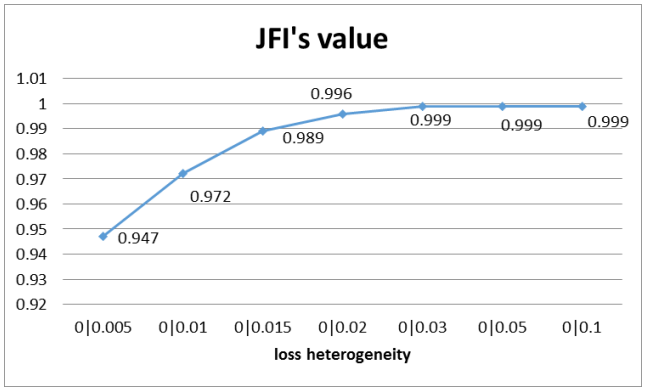

Fig. 12. Jain's fairness indexes in terms of unbalanced packet loss rate level of sub-flows.

From all of the experiments, we concluded that MPTCP's behavior was the same in both the homogeneous and heterogeneous networks. In the shared bottleneck link, the throughput of MPTCP was greater and more unfair than TCP's throughput in the good propagation condition, but it gradually became weaker as the propagation changed and became worse. (shown in Fig. 11 and Fig. 12) There are two reasons for this. First, the congestion control scheme was directly related. In this paper, we used LIA [7], which is the standard congestion control scheme for MPTCP. The mathematical equation of LIA is expressed as follows:

$$W_{r=}\begin{cases} w_r + \min\left(\frac{N_{bytes}*mss_r}{w_r}, \frac{\alpha*N_{bytes}*mss_r}{\sum_{r=0}^{n} w_r}\right), ACK \text{ event} \\ w_r * 0.5, \text{ loss event} \end{cases} \quad (2)$$

where α is a parameter that regulates the aggressiveness of the MPTCP connection to make it do no harm to the TCP flows, and it can be calculated by Equation (3):

$$\alpha = \sum_r W_r . \frac{max_r \frac{W_r}{srtt_r^2}}{\left(\sum_r \frac{W_r}{srtt_r}\right)^2} \quad (3)$$

where:

- w_r: Congestion window size of the r^{th} subflow
- N_{bytes}: Number of bytes acknowledged by receiver

- mss_r: Maximum size of segment on the r^{th} subflow
- $srtt_r$: Smoothed round trip time for the r^{th} subflow

From the equation for α, equation, we can see it is apparent that the value of α always will be smaller than the value of $\frac{1}{w_r}$ (increase index of Reno scheme). In homogeneous network, with high latency and/or high packet loss, α's value is low so that the increase of congestion window is slower and therefore the congestion window size is so small. Therefore, it makes the MPTCP's throughput is decreased. Additionally, the loss also will be more frequent and consequently MPTCP connection does not have more advances despite of multi-path. In the heterogeneous network, the first sub-flow was set to very high latency, and the second sub-flow and the TCP connection had the same, small latency value. MPTCP's performance can be approximated as follows:

$$B \approx \frac{W_1(\alpha)}{SRTT_1} + \frac{W_2(\alpha)}{SRTT_2} \approx \frac{W_2(\alpha)}{SRTT_2} \tag{4}$$

where $W_1(\alpha)$ and $W_2(\alpha)$ are the functions of the congestion window's size for first and second sub-flows, respectively.

In this case, variable $\alpha = \frac{W_2}{(W_2+\frac{SRTT_2}{SRTT_1}*W_1)^2} \leq \frac{W_2}{{W_2}^2} = \frac{1}{W_2}$ (Reno's increased index). This shows that, in this case, the MPTCP's performance will be less than the TCP's throughput, and our measured results were in agreement with this conclusion.

Another reason is caused by MPTCP's scheduler module in the Linux kernel. It prefers sending on the path with the lowest latency. It always fills up the sub-flow with lower RTT, then works with the next higher RTT. Therefore, in case of a large imbalance of the sub-flows, the bad link will have the lower priority to send data.

5. Conclusion

In this paper, we presented the MPTCP's performance over a shared bottleneck link. We concluded that the MPTCP's performance was outstanding in a normal network, such as a local area network that has a small latency and a small packet loss rate. However, in a high latency network or a highly-congested network, MPTCP's performance is not better than TCP's performance. For further study, we will continue examining the MPTCP's performance with other congestion control schemes and rebuild the congestion control schemes to solve their problems, as we did above.

References

[1] Raiciu, C., Handley, M., & Bonaventure, O. (January 2013). TCP extensions for multipath operation with multiple Addresses. *RFC 6824*.

[2] Paasch, C., & Barre, S. V. Multipath TCP implementation in the Linux kernel. From: http://www.multipath-tcp.org.

[3] Stewart, R. (2007). Stream control transmission protocol. *IETF RFC 4960*.

[4] Iyengar, J., Amer, P. D., & Stewart, R. R. (October 2006). Concurrent multipath transfer using SCTP multihoming over independent end-to-end paths. *IEEE/ACM Transactions on Networking (TON)*, *14(5)*, 951–964.

[5] Barre, S., Paasch, C., & Bonaventure, O. (May 2011). Multipath TCP: From theory to practice. *Proceedings of the 10th International IFIP TC 6 Conference on Networking*: *Vol. 1* (pp. 444-457).

[6] Raiciu, C., Handley, M., & Wischik, D. (October 2011). Coupled congestion control for multipath transport protocols. *RFC 6356*.

[7] TECHNO. Network Bandwidth Latency and Delay Simulation Tutorial. From: http://www.technogumbo.com/tutorials/Network-Bandwidth-Latency-and-Delay-Simulation-Tutorial/Network-Bandwidth-Latency-and-Delay-Simulation-Tutorial.php

[8] Mitchell B. Network Bandwidth and Latency. From http://compnetworking.about.com/od/speedtests/a/network_latency.htm

[9] Jain, R., Chiu, D., & Hawe, W. (1948). A quantitative measure of fairness and discrimination for resource allocation in shared computer systems. *Technical Report DEC-TR-301, Digital Equipment Corporation*, Maynard, Mass, USA.

Soonghwan Ro received B.S., M.S., and Ph.D. degrees from the Department of Electronics Engineering at Korea University in 1987, 1989, and 1993, respectively. He was a research engineer of the Electronics and Telecommunications Research Institute and University of Birmingham in 1997 and 2003, respectively. Since March 1994 he has been a professor at Kongju National University, Korea. His research interests include 5G communication, mobile network, and embedded systems.

Nguyen Van Dien received his B.S. degree from the Department of Electronics and Telecommunication at Hanoi University of Science and Technology in 2013, Hanoi, Vietnam.

From September 2013 to now, he studied for his M.S. degrees at the Department of Information and Telecommunication at Kongju National University. His research interests currently include computer science and network management.