African Journal of Engineering Research
Vol. 12(2), pp. 12-19, October 2024
ISSN: 2354-2160
DOI: 10.30918/AJER.122.24.012
Full Length Research Paper

Study on throughput-based congestion control of MPTCP and schedulers for WeBQoE improvement

Ennouhe Taleb*, Ito Yoshihiro and Takeshi Kato

[1]Nagoya Institute of Technology, Nagoya, Japan.

Accepted 9 October, 2024

ABSTRACT

This study proposes a new congestion control scheme for multi-path TCP to improve the user experience of Web services (WebQoE). It then studies its combination with a scheduler to suppress QoS fluctuations in MPTCP by experiment. The experimental results show that the proposed scheme suppresses throughput fluctuations, i.e., QoS fluctuations, better than existing MPTCP congestion control schemes in various environments, confirming the effectiveness of our proposal. The results also show that QoS fluctuations can be suppressed by combining a scheduler that suppresses RTT fluctuations with congestion control in specific environments.

Keywords: MPTCP, WebQoE, TCP, RTT, QoS.

*Corresponding author. E-mail: cmm14064@ict.nitech.ac.jp.

INTRODUCTION

Nowadays, electronic gadgets have multiple network interfaces and can connect to the Internet using 4G, 5G, or wireless LAN. However, TCP (Postel, 1981), the primary transport layer protocol in the TCP/IP protocol, can only handle one path per connection, making it impossible to use them effectively. Therefore, it is beneficial to create multiple paths at the same time. Thus, next-generation transport layer protocols such as SCTP (Stream Control Transmission Protocol) (Stewart, 2007) and MPTCP (Multi-Path TCP) (Ford et al., 2013) that can use multiple paths simultaneously are being standardized. SCTP is incompatible with TCP among such next-generation protocols, and its adoption will require much time and cost. On the other hand, MPTCP is an extension of TCP that uses TCP header options and is highly compatible with TCP. Therefore, we are focusing on MPTCP in this study. MPTCP enables the simultaneous use of multiple paths using multiple TCP flows, called sub-flows, for a single connection. This allows multiple paths and can improve the quality of service (QoS) compared to TCP. On the other hand, the QoS provided by MPTCP at the transport layer affects WebQoE (Quality of Experience for Web services). Muraki and Ito (2015) shows that, even if a high QoS is provided, high QoS fluctuation degrades WebQoE. Thus, a previous study (Noda and Ito, 2018) has proposed a new MPTCP congestion control scheme that suppresses QoS fluctuation to improve WebQoE. The method proposed in Noda and Ito (2018) uses RTT as a criterion of QoS, but either the throughput or the packet loss rate can also be considered parameters. The throughput especially significantly impacts WebQoE more than RTT because Web services may often send a large amount of traffic. In this paper, we first propose a new MPTCP congestion control that suppresses QoS fluctuations based on the throughput and then confirms its effectiveness through actual experiments. However, we know that the factors that significantly affect the QoS provided by MPTCP are congestion control (Welzl, 2005) and the scheduler that selects paths ("Multi-Path TCP," n.d.), the combination of which significantly impacts QoS. Second, this paper proposes a new congestion control scheme that uses the QoS of throughput as a parameter and suppresses the

fluctuation of QoS to improve WebQoE. It then investigates the effectiveness of congestion control combined with a scheduler. Secondly, this study treats the congestion control of the first part and investigates the combination with an optimal scheduler that can suppress QoS fluctuations by experiment.

MPTCP

Outline

MPTCP is one of the next-generation transport layer protocols. MPTCP connections can utilize multiple TCP flows, called sub-flows, simultaneously using multiple paths. Like TCP, MPTCP has window-based congestion control. MPTCP is being standardized as one of the next-generation transport layer protocols to solve the shortcomings of TCP. MPTCP uses multiple TCP flows, called sub-flows, allowing multiple routes to be used simultaneously. This will enable MPTCP to increase the available bandwidth compared to TCP, improve availability, and achieve a higher quality of service. Like TCP, MPTCP uses a three-way handshake to establish connections. However, MPTCP's three-way handshake is accompanied by the MPCAPABLE option to check whether the peer supports MPTCP. The communication is performed using standard TCP if the peer does not support MPTCP. Since MPTCP is a transport layer protocol, its congestion control significantly affects WebQoE 2 (Muraki and Ito, 2015). The current congestion controls for MPTCP can be categorized into loss-based and delay-based ones. For example, loss-based controls are LIA (Raiciu et al., 2011), OLIA (Khalili et al., 2011), and delay-based WVEGAS (Cao et al., 2012). The control proposed in this study is a delay-based one. MPTCP has two factors: congestion control and a scheduler, which work as follows: MPTCP receives data from an application, divides it into sub-flows, and distributes the divided data to each sub-flow using the scheduler. Each sub-flow sends the data distributed by the scheduler based on congestion control. In MPTCP, the scheduler and congestion control are closely related. For example, suppose the scheduler allocates data to a sub-flow whose congestion window size has been reduced by congestion control or to a sub-flow with no available congestion window. In that case, it may increase congestion or prevent efficient communication.

Congestion control

Like TCP, MPTCP has congestion control using a congestion window, which is variable. However, MPTCP's congestion control still differs from TCP's in that each sub-flow has its own congestion window. Currently, typical congestion control schemes for MPTCP include loss-based schemes such as LIA (Linked Increases Algorithm) (Raiciu et al., 2011), OLIA (Opportunistic Linked Increases Algorithm) (Khalili et al., 2011), and delay-based methods such as WVEGAS (Weighted VEGAS) (Cao et al., 2012).

Scheduler

When multiple sub-flows are available, MPTCP must select which sub-flow to send data to; the packet scheduler makes this selection. Currently, MPTCP has a default scheduler that determines the sub-flow with the smallest RTT, a round-robin scheduler that redundantly selects available sub-flows in order, and a redundant scheduler that selects the traffic of available sub-flows. Reference (Noda and Ito, 2019) also proposes a scheduler that suppresses RTT fluctuations and confirms its effectiveness by experiment.

PROPOSAL

This control is based on (Noda and Ito, 2018). This method uses the throughput instead of RTT as a QoS parameter and estimates it. We show our proposed control as follows: Firstly, a sender estimates the mean throughput after entering congestion avoidance (Stevens, 1997). Here, the following equation is used to estimate throughput:

$$\text{Throughput} = \frac{cwnd * MMS}{RTT} \quad (1)$$

Throughput is the estimated throughput, cwnd is the congestion window size, MSS is the maximum segment size, and RTT is the time between sending a segment and receiving an ACK. Secondly, the estimated throughput is compared with the mean. If Throughput is greater than the mean, the cwnd is decreased. And if Throughput is less than the mean, the cwnd is increased. The cwnd is controlled by:

$$cwnd = \begin{cases} cwnd - \alpha \\ (Throughput > Mean_{Th} + \gamma \times Mean_{Th}) \\ cwnd + \beta \\ (Throughput < Mean_{Th} - \gamma \times Mean_{Th}) \end{cases} \quad (2)$$

However, two main points can be noted: we can confirm that TSN standards helped improve the overall QoS of the network when implemented, as all the QoS-controlled traffic has lower latency except for one (Traffic 2). Where α and β represent the amount of increase and decrease for cwnd, respectively, and γ adjusts the degree of throughput fluctuation. If we increase cwnd, throughput will be higher; if we decrease cwnd, throughput will be lower. Moreover, $Mean_{Th}$ indicates an estimation of the mean

throughput. The user specifies the values of α and β. By changing α and β, it is possible to adjust the degree of congestion window control for each communication according to the type of service.

EXPERIMENTS

Outline

Two experiments are conducted. They are referred to as Experiment A and Experiment B, respectively. Experiment A tests the MPTCP congestion control method by measuring the throughput and RTT in different environments. The environments are referred to as Env1 through Env6. Experiment B combines the abovementioned MPTCP's congestion control method with a scheduler and studies the outcome.

Experiment A

Figure 1 indicates the network environment of Experiment A. Note that Experiment B also uses the same network environment. It consists of a Web server, a Web client, three network emulators, and two pairs of Load servers and Load clients. As a target Web service, we adopt a map service widely used worldwide, such as Google Maps. The Network Emulator adds delay and packet losses to packets that pass through it, thus creating an environment where the communication quality of each route is uniform and heterogeneous. The subject is accessing Web service through a Web client. The Web server and Web client are connected via a network emulator. Table 1 shows the parameter values of the network emulators. Since this study aims to confirm whether the proposed method can suppress throughput fluctuations in any environment, we also check the necessary values to suppress the fluctuation by changing the values of α and β. Table 2 shows that we also congest each route by generating TCP traffic between the Load client and the Load server. In this experiment, we confirm the effectiveness of the proposed method by comparing it with existing MPTCP congestion controls, such as LIA, OLIA, and WVEGAS, the proposed method, and the congestion control defined in Noda and Ito (2018). The QoS parameters to be measured are throughput and RTT.

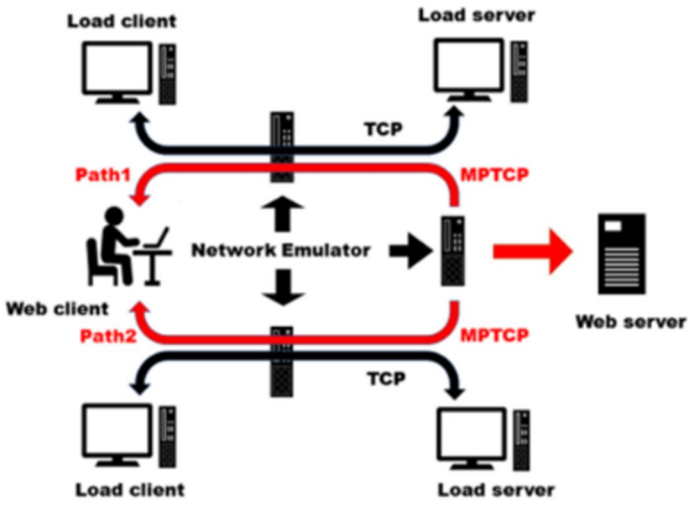

Figure 1. Experimental environment.

Table 1. Parameters values of network emulators.

	Bandwidth	Packet loss rate	Delay
Env1, Env2, Env3			
Path1	100 Mb/s	3%	50 ms
Path2	100 Mb/s	3%	50 ms
Env4, Env5, Env6			
Path1	100 Mb/s	1%	100 ms
Path2	100 Mb/s	3%	50 ms

Table 2. Number of TCP connections.

Env1	Env2	Env3	Env4	Env5	Env6
10	20	10 ~20	10	20	10 ~20

Result of Experiment A

Figures 2 and 3 show the results of Experiment A. In these figures, the abscissa plots the experimental environment, and the ordinates indicate the variance of the throughput, the mean throughput, the variance of the RTT, the mean RTT, and the mean of the RTTs. From Figure 2(a), we see that the proposed method can suppress the throughput fluctuation compared to existing MPTCP congestion controls in environments where the communication quality of each route is uniform. Moreover, we also find that the throughput is kept low in Figure 2(b). A trade-off exists

between improving QoS and suppressing QoS fluctuations in Noda and Ito (2018). Therefore, the proposed scheme can also suppress throughput fluctuation by keeping the throughput low. Also, by adjusting α and β in Equation 2, congestion control can be performed with appropriate control for the communication volume, thereby controlling the fluctuation. For example, in environment 1, setting α to 1 and β to 5 will lead to more suppression of fluctuations. Similarly, it was found that throughput fluctuations can be suppressed in heterogeneous environments with different communication quality. From Figure 3, we recognize that the proposed method also suppresses RTT fluctuations compared to existing MPTCP congestion control methods in an environment where the communication quality of each route is uniform. The RTT is kept low, indicating a trade-off between the average RTT and the suppression of RTT fluctuation. Similarly, the proposed scheme can suppress RTT fluctuations even in heterogeneous environments with different communication quality, compared to the congestion control scheme in Noda and Ito (2018), which suppresses RTT fluctuations.

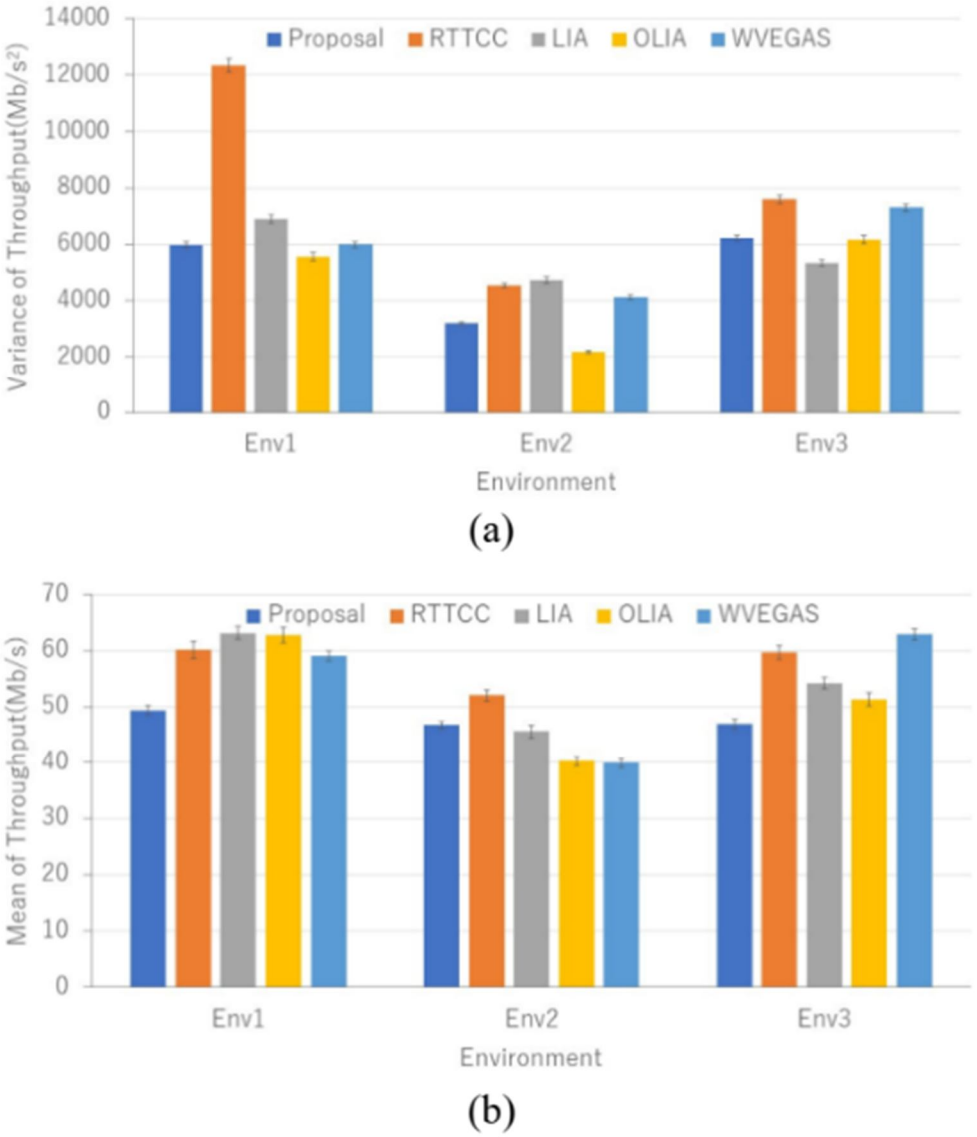

Figure 2. Variance and mean throughput from Env1 to Env3.

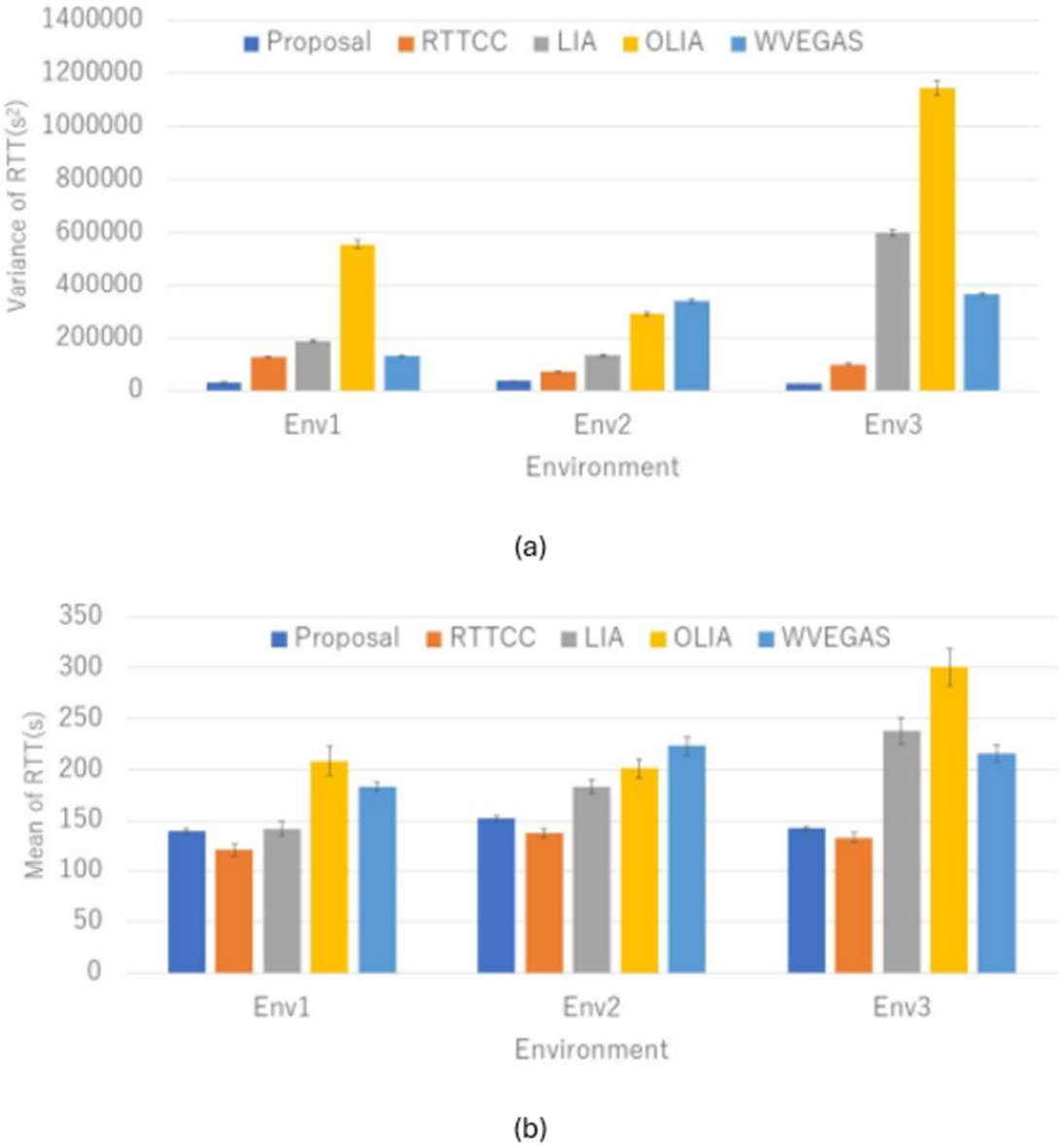

Figure 3. Variance and mean of RTT from Env1 to Env3.

Experiment B

This experiment treats the congestion control of Experiment A and investigates its combination with an optimal scheduler that can suppress QoS fluctuations. In the experimental environment, two paths are provided between the Web client and the Web server for MPTCP communication. The network emulator acts as a router and changes the communication quality of each route by adding delay and packet losses to packets passing through it. In addition, the network emulator creates a uniform environment in which the quality of communication is the same for each route and various environments in which the quality of communication is different for each route. These six environments are shown in Table 1. For convenience, the experimental environment is named Env1 through Env6. The network emulator used is Dummynet (Rizzo, n.d.). The load clients and load servers send and receive TCP traffic to congest each route. The traffic is handled by a benchmark application, auto-bench ("Autobench," n.d.). The experiment uses the map search service ("Google," n.d.) as the Web service. For congestion control, the Web server uses the congestion control of the first study, and the Web client utilizes LIA,

which is a standard for MPTCP congestion control. In this experiment, we employ four schedulers: a default scheduler, a round-robin scheduler, a redundant scheduler, and one similar to the scheduler from Noda and Ito (2019). This study defines these schedulers as default, round-robin, redundant, and RTTS.

Result of Experiment B

Figures 4 through 7 show the results of Experiment B. In each figure, the abscissa represents the environment. The ordinates in Figures 4 and 5 indicate the standard deviation of throughput, while those in Figures 6 and 7 represent the mean of throughput.

First, Figures 6 and 7 indicate the standard deviation of the throughput when congestion control is combined with multiple packet schedulers. Figures 4 and 5 show that RTTS exhibits the lowest throughput variability in all six environments compared to the existing MPTCP packet scheduler. This means that throughput fluctuations are reduced by using RTTS in all environments. Consequently, the congestion control proposed in the first study can suppress throughput fluctuations more when used with the schedule (Noda and Ito, 2019). This is because the scheduler suppresses RTT fluctuations, reducing the variation in the number of packets received in a unit of time, resulting in less throughput variation. The results of the mean throughput are plotted in Figures 5 and 6. These results mean that throughput is lower when the scheduler is RTTS. The results of the variance and the mean of the throughput decrease with the reduction of the variance of the throughput, suggesting a trade-off between the reduction of the variance of the throughput and the mean of the throughput. RTTS preferentially selects the sub-flow with the least congestion to transmit data, resulting in lower throughput. The above confirms that congestion control that suppresses fluctuations in throughput can be combined with a scheduler that suppresses fluctuations in RTT rather than an existing packet scheduler to suppress fluctuations in QoS. The lower throughput indicates a trade-off between suppressing the fluctuation of the throughput and the mean of the throughput. This means that while it is necessary to keep the throughput low to suppress the fluctuation of the throughput, a very low throughput may lead to a decrease in WebQoE. Consequently, it is essential to evaluate QoE through actual subjective experiments, congestion controls, and schedulers that can control the throughput so that it does not become too low while suppressing the fluctuation of the throughput.

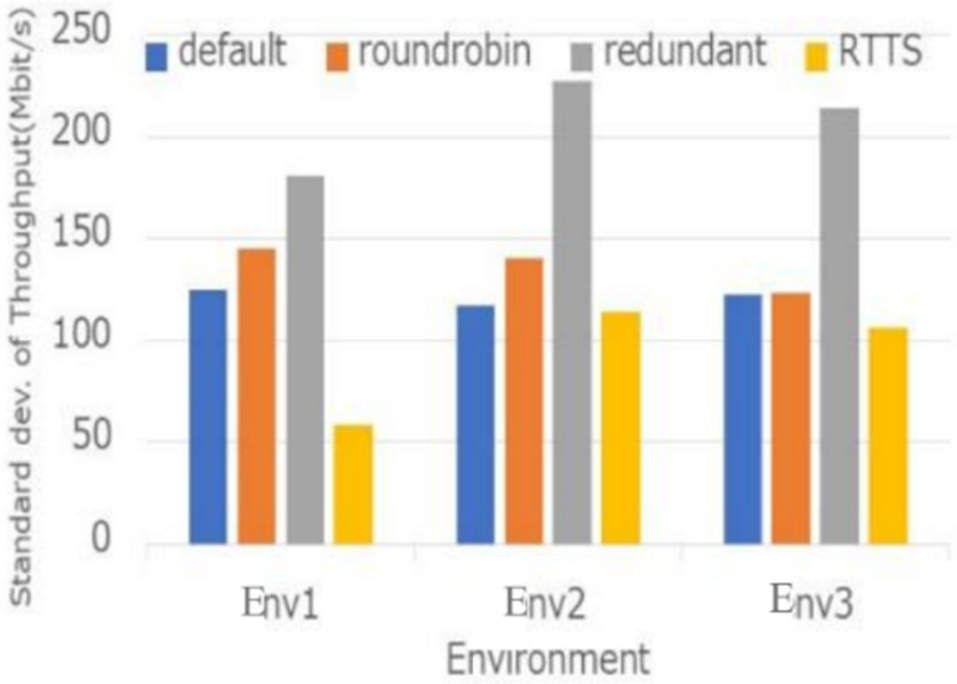

Figure 4. Standard deviation of throughput (Env1 through Env3).

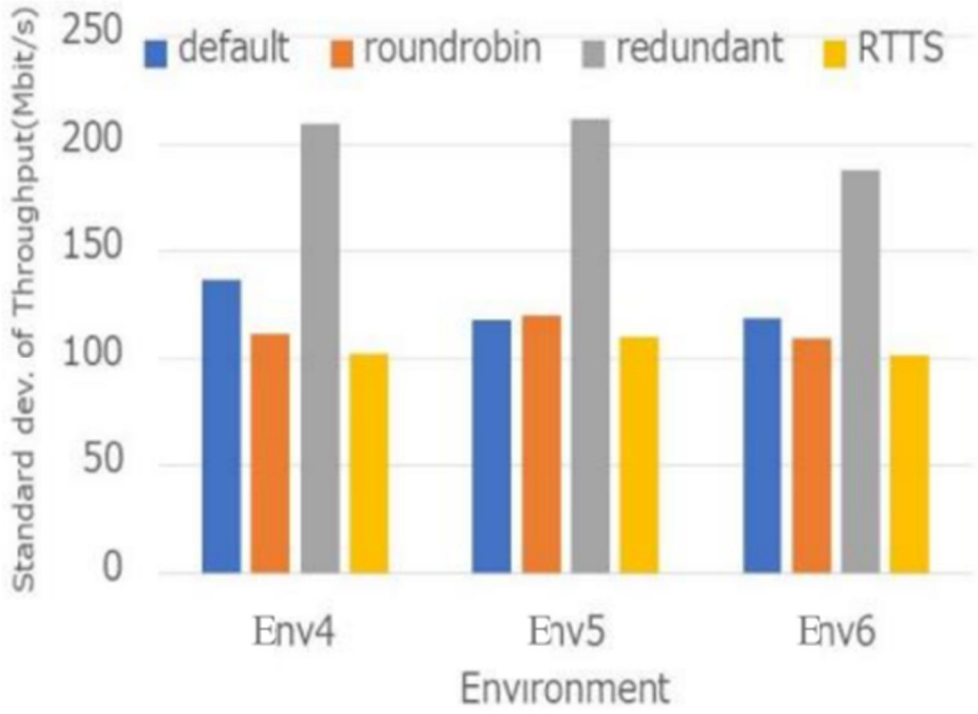

Figure 5. Standard deviation of throughput (Env4 through Env6).

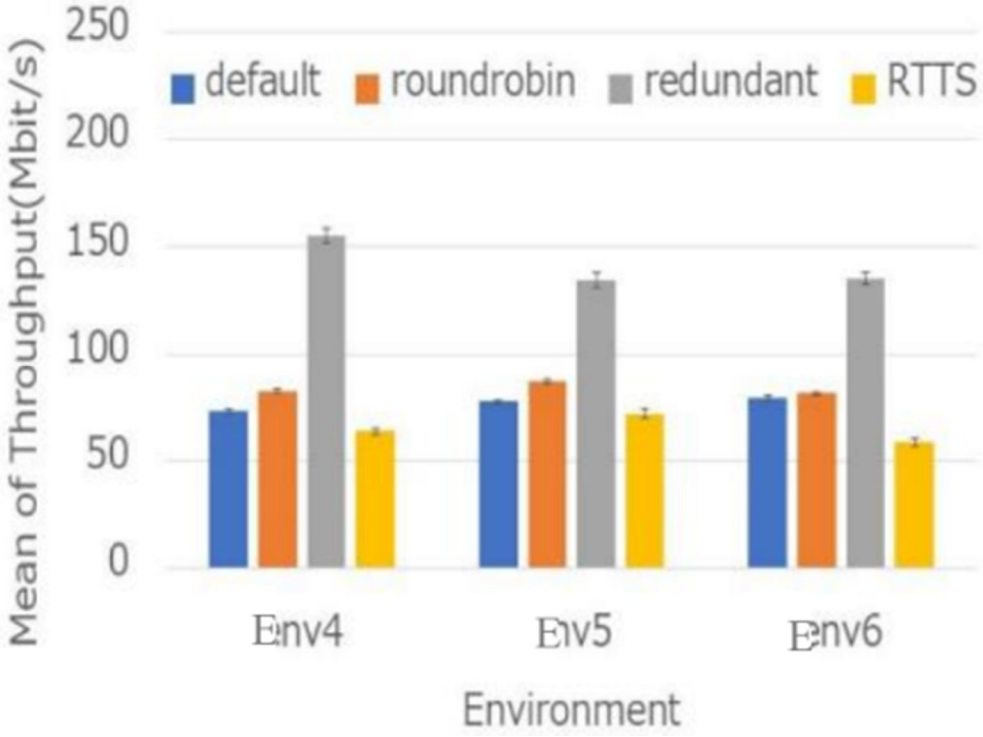

Figure 6. Mean of throughput (Env.1 through Env3).

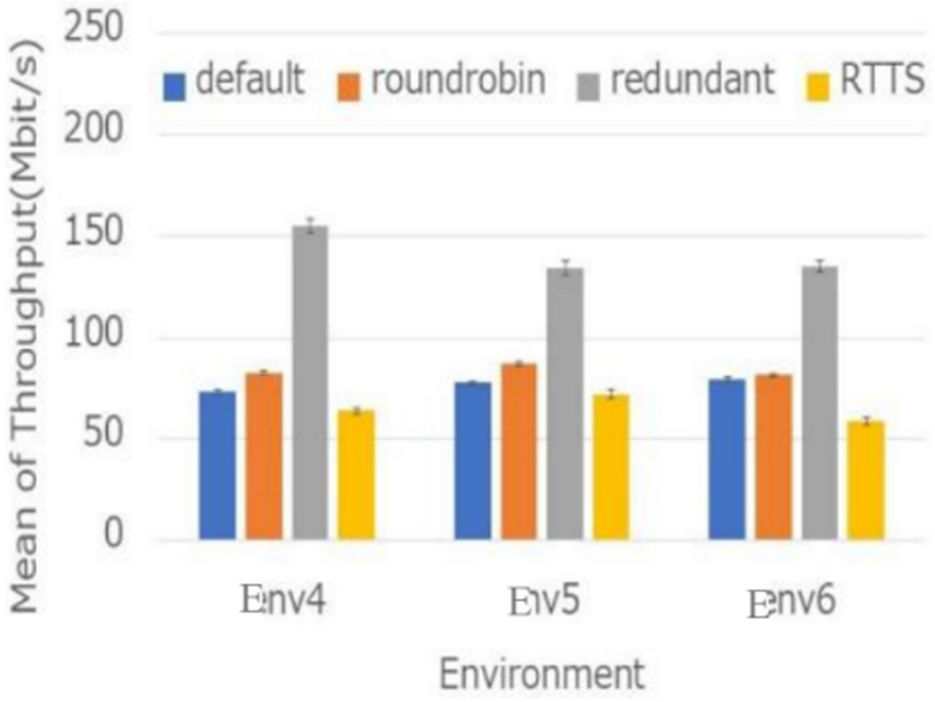

Figure 7. Mean of throughput (Env.4 through Env6).

CONCLUSION

This study proposed a new MPTCP congestion control that suppresses QoS fluctuations based on throughput instead of RTT. We evaluated the QoS through experiments, and the experimental results proved the effectiveness of our proposal by showing that the proposed method can suppress throughput fluctuation by keeping the throughput low and suppressing throughput fluctuations in heterogeneous environments. Furthermore, we studied which packet scheduler can suppress QoS fluctuation by combining that congestion control with experimentation. The experimental results show that a congestion control that suppresses the fluctuation of the throughput can suppress QoS fluctuation combined with a scheduler that suppresses RTT fluctuations rather than an existing packet scheduler to suppress fluctuations in QoS. It was also observed that a low throughput can suppress the fluctuation; too low throughput can decrease the Quality of the Web user's Experience. Therefore, in our future work, we will assess QoE and confirm whether the combination shown in this study improves WebQoE. We will also evaluate QoS in various environments and services.

REFERENCES

Autobench (n.d.). http://www.xenoclast.org/autobench/

Cao, Y., Xu, M., & Fu, X. (2012). Delay-based congestion control for multipath TCP. In *Proceedings of the IEEE 20th International Conference on Network Protocols (ICNP)* (pp. 1-10).

Ford, A., Raiciu, C., Handley, M., & Bonaventure, O. (2013). TCP extensions for multipath operation with multiple addresses. *IEEE RFC 6824 (Experimental)*.

Google. (n.d.). https://www.google.co.jp/maps.

Khalili, R., Gast, N., Popovic, M., Upadhyay, U., & Le Boudec, J.-Y. (2011). MPTCP is not Pareto optimal: Performance issues and a possible solution. In *Proceedings of the 8th International Conference on Emerging Networking Experiments and Technologies* (pp. 1-12). New York, NY, USA.

Multi-Path TCP – Linux Kernel Implementation. (n.d.). http://multipathtcp.org

Muraki, Y., & Ito, Y. (2015). Study on effect of congestion control of multipath TCP on WebQoE. In *Proceedings of the IEEE 4th Global Conference on Consumer Electronics (GCCE)* (pp. 52-53). Osaka, Japan.

Noda, K., & Ito, Y. (2018). Proposal of novel MPTCP congestion control to suppress QoS fluctuation for WebQoE improvement. In *Proceedings of ICCE Berlin 2018*.

Noda, K., & Ito, Y. (2019). Proposal of multi-path TCP packet scheduler to adjust trade-off between QoS fluctuation and throughput for WebQoE improvement. In *Proceedings of the IEEE 4th International Conference on Computer and Communication Systems (ICCCS)* (pp. 493-496).

Postel, J. (1981). Transmission control protocol. *IEEE RFC 793 (INTERNET STANDARD)*.

Raiciu, C., Handley, M., & Wischik, D. (2011). Coupled congestion control for multipath transport protocols. *IEEE RFC 6356 (Experimental)*.

Rizzo, L. (n.d.). Dummynet. http://info.iet.unipi.it/luigi/dummynet/

Stevens, W. R. (1997). TCP slow start, congestion avoidance, fast retransmit, and fast recovery algorithm. *RFC 2001*.

Stewart, R. (2007). Stream control transmission protocol. *IEEE RFC 4960 (Proposed Standard)*.

Welzl, M. (2005). *Network congestion control: Managing internet traffic* (W. S. Kingdom, Ed.). John Wiley and Sons.