MPTCP Evaluation for Heterogeneous Wireless Networks

Naotaka Sakaguchi[1] *Member,* Takenori Sumi[1] *Senior Member*, Yukimasa Nagai[1] *Senior Member*,
Jianlin Guo[2] *Senior Member*, Kieran Parsons[2] *Senior Member*, and Philip Orlik[2] *Senior Member*,
[1]Information Technology R&D Center, Mitsubishi Electric Corporation
5-1-1 Ofuna, Kamakura, Kanagawa, 247-8501 JAPAN
Sakaguchi.Naotaka@bk.MitsubishiElectric.co.jp

[2]Mitsubishi Electric Research Laboratories Inc.
201 Broadway Cambridge, Massattusets 02139 USA

***Abstract*— In recent years, more and more wireless devices are equipped with multiple communication interfaces. From the perspective of increasing communication capacity and stability through effective use of frequencies, multi-path communication technology using multiple interfaces simultaneously has been attracting attention. Multi-Path TCP (MPTCP), which is an extension of TCP/IP protocol widely used today, has been studied extensively worldwide. We implemented the MPTCP specification in ns-3, an open-source network simulator, and evaluated its behavior by comparing its performance with that of a Linux PC. As a result, we found that our simulator behaves similarly to Linux PC in an environment without packet loss. On the other hand, when there is packet loss, the behavior of the simulator is different from that of Linux PC. This paper reports our simulation environment and evaluation results.**

Keywords—MPTCP, IoT, flexible wireless networks, ns-3

I. Introduction

Many wireless devices, such as smartphones, commonly have multiple wireless interfaces, such as LTE and wireless LAN. However, using conventional TCP/IP specifications, only one path can be chosen for communication. Therefore, these devices follow certain rules to select only one interface to set up communication path, e.g., if both LTE and wireless LAN are available, devices choose wireless LAN. Due to this nature, if a high-priority path is unstable, the communication quality will be affected. In recent years, IETF is standardizing the Multi-Path TCP (MPTCP) specification, which has been attracting attention because it can effectively use frequencies by utilizing multiple paths simultaneously, and it is expected to stabilize communications and improve throughput. In this study, we implemented the MPTCP functions in ns-3[1], an open source software (OSS) network simulator, and evaluated its operation and performance in comparison with the results obtained on a real Linux machine. Our objective is to provide a robust MPTCP simulator that can be used by researchers and academia for convenience.

The rest of this paper is organized as follows. Section 2 presents MPTCP overview. Related work for MPTCP is provided in Section 3. Performance evaluation set up is described in Section 4. Section 5 introduces the performance results. Finally, we conclude our work in Section 6.

II. MPTCP Overview

MPTCP is an extension to TCP/IP defined in RFC8684 [2]. MPTCP enables multi-path communications by simultaneously using multiple communication interfaces. The conceptual diagram of MPTCP connection establishment method and structure is shown Figure 1. At first, devices perform 3-way-handshake using primary path. At this time, the option field of these messages are set to *MP_CAPABLE*. Once the connection between the main flows is established and data packet transmission starts, the sub-flow is added. At this time, the optional *MP_JOIN flag* is set. This procedure is repeated to establish the number of available sub-flows. Figure 2 shows data transmission scheme using 3 MPTCP paths. The MPTCP meta-socket receives data from the application and assigns it a sequence number for the receiving device to reconstruct the data. The scheduler then distributes the data to flow paths. Each data distributed to a specific path by the scheduler is labeled a sequence number for normal TCP communication. By assigning a sequence number for TCP communication and data reconstruction for each path, it becomes possible to reconstruct the original data even if data packets arrive out of order. The specific functions of the aforementioned schedulers depend on the implementation and are not defined by the standard. Below are some typical schedulers:

Default : The flow with the smallest RTT (Round Trip Time) is selected as a priority. When the window size of the highest priority path is full, data is distributed to another path with the next smallest RTT.

Round-robin : Assign one packet to each path in turn. This is basically for research purposes and is deprecated.

Redundant : To improve the data reachability, the same packets is assigned to all paths, and achieve the redundancy.

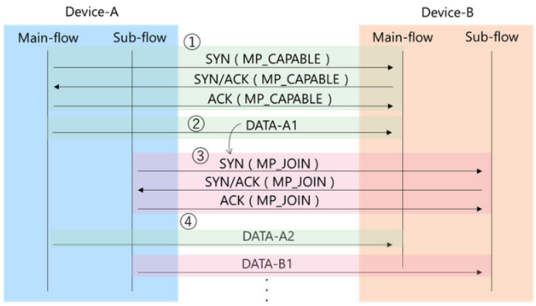

Figure **1**: MPTCP Connection Establish

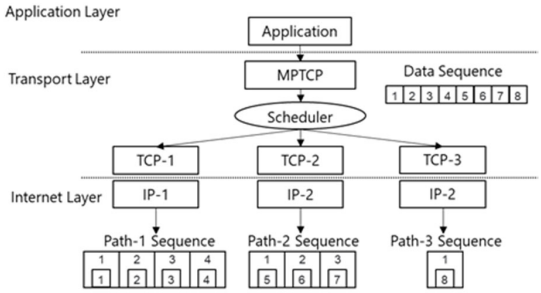

Figure **2**: Data Transmission Scheme

III. RELATED WORK

Performance evaluations using real equipment have been conducted for both wired and wireless systems, and both advantages and disadvantages have been reported in [4][5][6]. These evaluations have confirmed improved throughput and stability in file transfers with large data sizes. In particular, there is room for improvement in the scheduling function. In the case of multi-link communication, when there is a difference in delay between paths or when the delay is asymmetric, such as in LTE and WLAN, the order of received packets changes and "out-of-order" occurs. To solve this problem, W. Yang et al. [7] and Y. Xing et al. [8] developed schedulers that can properly assign data to each path based on RTT and window size, respectively. There is also interest in congestion control algorithms that can be combined with MPTCP. I. Mahmud et al. [9] evaluated the performance of different congestion control algorithms using real equipment. The following algorithms are widely used in the evaluation of matching with congestion control algorithms.

NewReno : Old typical loss-based algorithm. Starts transmission with a small window size at the beginning of communication (slow start), and exponentially increases the window size each time an ACK is received. If packet loss has occurred, the window size is reduced by half. Thereafter, the window size is increased linearly.

Cubic : Congestion control algorithm that is currently the default setting for Linux. Initially, the window size is increased exponentially with a "slow start". If packet loss occurs, the window size is reduced to about 70%, after which the window size increases cubically. Since the amount of increase is determined by the window size and the time since packet loss occurred, a large RTT tends to slow the rate of window size increase.

BBR(Bottleneck Bandwidth and Round-trip propagation time) : Delay-based congestion control algorithm developed by Google in 2016 [10]. It uses RTT as an indicator of congestion occurrence; by monitoring RTT, it detects deteriorating communication conditions before packet loss occurs and changes the window size. It is attracting attention as a congestion control algorithm with high bandwidth fairness. In particular, W. Wei et al. [11] developed a scheduler for BBR and evaluated its performance both in simulation (ns-3.3) and on actual machines and reported improved throughput.

In ns-3, traffic control and queueing have been reworked for Linux Kernel since ns-3.25. *Reno* congestion control algorithm has been added since ns-3.32 and *Cubic* has been added since ns-3.33. Furthermore, from ns-3.34, IEEE802.11ax function and *BBR* congestion control algorithm were implemented. In particular, ns-3.3 enables carrier aggregation and LTE handover, and improves the performance of the 4G/5G MIMO model. For these reasons, the version used by W. Wei et al. [11] may not be able to simulate the latest wireless standards or other communication standards. K. Nadeem et al. [12] created and evaluated an implementation of MPTCP based on ns-3.29. However, due to the reasons discussed in the previous paragraph, their implementation was found to be difficult to simulate current Linux machines and the latest wireless protocols. Additionally, their implementation had issues such as sending all packets from the primary path if there are more than two paths, and not setting up a one-to-one path. Moreover, the default scheduler of Linux did not function properly, and sub-flows with narrow bandwidth were not used for communication. Therefore, they evaluated the implementation using Round-robin, which is not recommended in Linux. Hence, we modified their implementation, implemented MPTCP in ns-3.36, and present a new simulator that supports *BBR* and the latest wireless standards. We then evaluated its performance using typical loss-based congestion control algorithms, *NewReno* and *Cubic*, as well as the latest delay-based congestion control algorithm, *BBR*.

IV. EVALUATION SETUPS

This evaluation was conducted both using ns-3 simulatorand real Linux machines. In the simulation, ns-3.36 was implemented on Ubuntu 21.04 (Linux Kernel 5.11) in a virtual environment (VMware) for evaluation. The same network was constructed on a PC implementing Ubuntu 22.04 LTS (Linux Kernel 5.15) on the actual machine. The network configuration for this evaluation is shown in Figure 3. The transmitter (client) transmitted data for 60 seconds and its throughput was measured at the receiving end (server). The sending node has two communication interfaces, and their multipaths are bundled by an intermediate router and forwarded to the receiving side. The default settings for each path are 10 Mbps bandwidth and 10 ms delay. The evaluation was performed by varying the bandwidth of net2 (1Mbps/10Mbps), the packet loss rate of net1 (0%/0.1%), and the congestion control algorithm (NewReno/Cubic/BBR). The path from the router to the receiving node was set to high speed and low latency (bandwidth: 1 Gbps, latency: 0.1 ms)

so that buffer overflows in the router did not need to be considered.

V. RESULT & DISCUSSION

The results of the performance evaluation for the following conditions are shown in the figures below:

- ***Case 1***: no packet loss, no bandwidth variation
- ***Case 2***: no packet loss, with bandwidth variation (10Mbps/1Mbps)
- ***Case 3***: 0.1% packet loss, no bandwidth variation

The evaluation results show that all congestion control algorithms behave similarly in both the real device and the simulator when no packet loss occurs. When there is a 10-times difference in bandwidth, *BBR's* performance is superior, possibly due to its ability to adjust the window size based on RTT and mitigate packet queuing caused by bandwidth differences. In the case of packet loss, the throughput of Cubic degraded significantly in the simulator. It is possibly due to differences in the MPTCP scheduler functions between the simulator and the real machine, leading to retransmissions in Cubic congestion control algorithm. Fig. 4 demonstrates that without packet loss and with higher bandwidth, all algorithms obtain much higher throughput compared to the lower bandwidth case shown in Fig. 5 and the packet loss case in Fig. 6.

VI. CONCLUSION

From the perspective of effective use of frequencies, there have been many studies on the application of MPTCP to wireless communications, but most of them were evaluated in real equipment, and even in the case of simulations using ns-3, they were based on older versions. Therefore, with a view to applying MPTCP to the latest wireless communication standards in the future, we implemented MPTCP specifications into ns-3.36 and evaluated its performance. As a result, we confirmed that our simulator behaves like the real device in the situations without packet loss. However, in situations with packet loss, *Cubic* throughput of the simulator degrades significantly. We presume that this issue is caused by a mismatch between the algorithm for the retransmission function in *Cubic* and the MPTCP scheduler in the simulator. Therefore, analysis of the interrelationship between the *Cubic* algorithm and the MPTCP functional block is a future work.

REFERENCES

[1] ns-3 Network Simulator, https://www.nsnam.org/.

[2] A. Ford, C. Raiciu,M. Handley, O. Bonaventure, and C. Paasch, "TCP extensions for multipath operation with multiple addresses," 2020, RFC 8684, Accessed: Oct. 3, 2022, https://www.ietf.org/rfc/rfc8684.txt.

[3] Multipath TCP – Linux Kernel implementation, https://www.multipath-tcp.org/.

[4] Y.Kinoshita, I.Arai, K.Fujikawa, "A Proposal of MPTCP Scheduler considering one-way latency information and out-of-order under various delay conditions", ShingakuGihou, vol.118, no.360, pp.59-63, May 2018.

[5] S. C. Nguyen, X. Zhang, T. M. T. Nguyen, and G. Pujolle, "Evaluation of throughput optimization and load sharing of multipath TCP in heterogeneous networks". "2011 Eighth International Conference on Wireless and Optical Communications Networks".

[6] L. E. B. Wiesse, H. F. C. Condori, and A. A. Mendoza, "Design and Implementation of a Network Based on Multipath TCP of Local Production", "2018 IEEE XXV International Conference on Electronics, Electrical Engineering and Computing (INTERCON)".

[7] W. Yang, P. Dong, L.Cai, and W. Tang, "Loss-Aware Throughput Estimation Scheduler for Multi-Path TCP in Heterogeneous Wireless Networks", IEEE Transactions on Wireless Communications, Volume: 20, Issue: 5, May 2021.

[8] Y. Xing, K. Xue, Y. Zhang, J. Han, J. Li, J. Liu,and R. Li, "A Low-Latency MPTCP Scheduler for Live Video Streaming in Mobile Networks", IEEE Transactions on Wireless Communications, Volume: 20, Issue: 11, November 2021.

[9] I. Mahmud, T.Lubna, Y.Song, and Y.Cho, "Coupled Multipath BBR (C-MPBBR): A Efficient Congestion Control Algorithm for Multipath TCP",IEEE Access, Volume:8, September 2020

[10] N. Cardwell, Y. Cheng, C.S. Gunn, S. H. Yeganeh, and V. Jacobson, "BBR: Congestion-Based Congestion Control", ACM Queue, vol. 14, September-October (2016), pp. 20 – 53.

[11] W. Wei, K. Xue, J. Han, Y. Xing, D. S. L. Wei, and P. Hong, "BBR-Based Congestion Control and Packet Scheduling for Bottleneck Fairness Considered Multipath TCP in Heterogeneous Wireless Networks", IEEE TRANSACTIONS ON VEHICULAR TECHNOLOGY, VOL. 70, NO. 1, JANUARY 2021.

[12] K. Nadeem and T. M. Jadoon, "An ns-3 MPTCP Implementation", Proc. 14th EAI International Conf. on "Quality, Reliability, Security and Robustness in Heterogeneous Systems.", pp48-60, Ho Chi Minh City, Vietnam, December 2018.

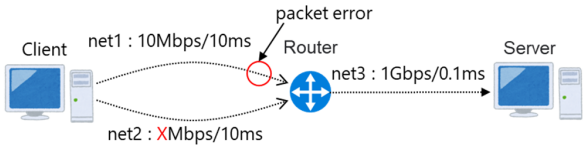

Fig. 3 Network Setup

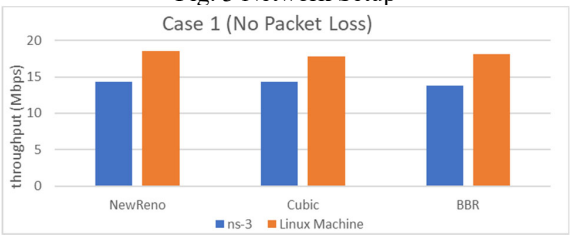

Fig. 4 Case 1 - Throughput without packet loss

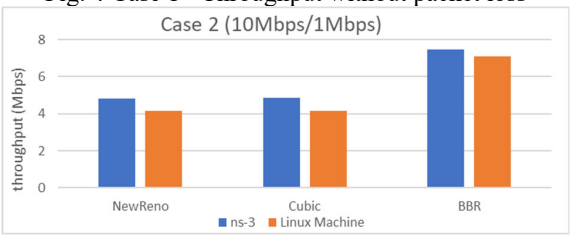

Fig. 5 Case 2 - Throughput with bandwidth variation

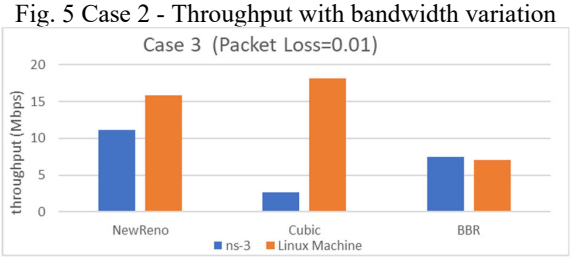

Fig. 6 Case 3 - Throughput with packet loss