Performance of Multipath TCP Schedulers in Concurrent Use of 5G and 4G Networks

Imtiaz Mahmud
School of Electronics and Electrical Engineering
Kyungpook National University
Daegu, Korea
imtiaz.tee@gmail.com

You-Ze Cho
School of Electronics and Electrical Engineering
Kyungpook National University
Daegu, Korea
yzcho@ee.knu.ac.kr

***Abstract*—Currently, mobile operators are deploying the 5G cellular network worldwide. Due to limited coverage and small-scale deployment, handoffs often occur from 5G to 4G and vice versa. These handoffs often lead to significant delays caused by retransmissions. Instead of using either 5G or 4G connection, simultaneous use of 5G and 4G, i.e., multipath connection, is anticipated to yield better performance. In this paper, we empirically study the performance of multipath TCP schedulers in such a scenario. Through NS3-DCE simulation experiments, we investigated the throughput, received SINR, and flow completion time for considered schedulers using the default Linux setup. We found that the BLEST scheduler shows the best performance because of its fast awareness and adaptability towards the head of line blocking.**

Keywords—MPTCP, MPTCP Scheduler, 5G, 4G, BLEST, ECF, Round-robin, Redundant

I. Introduction

The fifth-generation (5G) cellular network has recently attracted significant attention. 5G can provide much higher bandwidth with ultra-low latency than the fourth generation (4G) cellular network [1]. It contains some unique features, such as new core network management, new radio frequencies, massive MIMO, softwarization, etc. The 5G cellular network will soon be deployed worldwide and is estimated to bring business worth 13.2 trillion US dollars globally [2].

Currently, 5G has been commercially available on a small scale in a handful of countries worldwide. Due to the lack of full 5G coverage, frequent handoffs from 5G to 4G and vice versa occur for mobile user equipment (UE). This often causes packet drops and delays in data delivery. According to previous research, the transmission control protocol (TCP) suffers greatly from this kind of handoff, often resulting in long delays and low throughput compared to regular operation. The key reason behind this low performance is the spurious retransmission timeouts (RTOs) caused by the handoffs [3].

With the advancement of modern technology, most of the current mobile devices come with multiple communication interfaces. Notably, 5G-enabled mobile devices can operate under the 4G network too. On the other hand, the mobile operators are deploying 5G mostly alongside their existing 4G infrastructure. In many cases, even the 5G and 4G antennas are installed at the same tower in a non-standalone setup [4], as shown in Fig. 1. Moreover, following Qualqum's proposal in [5], instead of using either a 5G or 4G network at a time, simultaneous use of both the networks are expected to provide a better performance in terms of throughput and help overcome the existing problems, especially when the full worldwide deployment of 5G network still needs a significant amount of time [6]. A considerable study is essential before such a large implementation to apprehend the performance improvement, possible issues, and implementation of suitable protocols. Multipath TCP (MPTCP), a transport layer protocol that enables the simultaneous use of multiple communication interfaces, is suitable for such scenarios. However, proper research on MPTCP is still unavailable considering such scenarios. The only available study by Lan et al. gives a simple performance analysis of the overall performance of MPTCP.

This research was supported in part by the Basic Science Research Program through the National Research Foundation of Korea (NRF) No. NRF-2018R1A6A1A03025109 and No. NRF-2019R1A2C1006249.

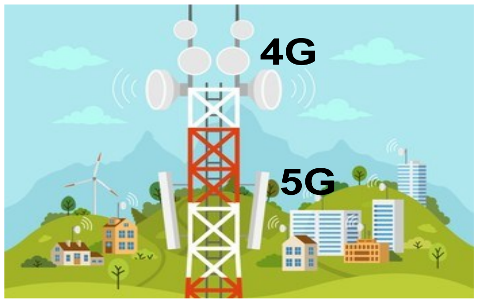

Fig. 1. Illustration of the non-standalone 5G implementation model: a 5G and 4G base stations are colocated at the same tower.

Therefore, in this paper, we focus on observing the performance of MPTCP schedulers in a scenario where a mobile UE is on the move and simultaneously uses both the 5G and 4G networks via an MPTCP connection. In MPTCP, each TCP flow going through the paths between a sender and a receiver is considered as a subflow (SF). And an MPTCP scheduler is responsible for selecting a suitable SF for sending each packet. The key contributions of this work can be listed followingly:

- We set up a simulation testbed integrating NS3-DCE [7] with the NS-3.33 mmWave [8] module and MPTCP v0.90 [9]. Alongside, we implemented schedulers such as Redundant, earliest completion first (ECF) [10], and blocking estimation (BLEST) [11] for a fruitful comparison.
- We compared throughput and flow completion time (FCT) and found that BLEST performs the best, thanks to its intelligent head-of-line (HoL) blocking prediction and preventing mechanism.

The rest of the paper is organized as follows: Section 2 gives a brief overview of the tested MPTCP schedulers, Section 3 details the simulation setup, Section 4 analyzes the performance, and Section 5 concludes the paper.

II. Overview of the Considered Schedulers

We tested widely accepted five MPTCP schedulers implemented in different versions of the MPTCP Linux kernel over time. This section gives a summary of the working procedure of those schedulers.

A. Shortest RTT first scheduler (SRTT)

It sends a packet through an SF with the lowest round-trip-time (RTT) among all SFs whose congestion window (CWND) is not full yet. If more than one path fulfills this criterion, the sender systematically chooses one SF and sends the packets via that SF until that SF's CWND becomes full. It will again come back to this path once the space CWND space becomes available again. It is currently the default scheduler in the MPTCP Linux kernel [12].

B. Round-robin

It selects an SF in a round-robin fashion. It picks up an SF one after the other circularly among all the available SFs whose CWND is not full yet [12]. All the SFs have an equal probability of being used for sending a packet

C. Redundant

The Redundant scheduler sends the same data packets through all the available SFs whose CWND is available. This approach ensures robustness against packet losses and helps reduce HoL-blocking [13]. However, it significantly wastes the available resources by putting extra pressure on the network to handle the same redundant data by all the network interfaces.

D. Earliest Completion First (ECF)

It makes a scheduling decision based on the amount of available data to be sent, the RTT, and the available CWND of an SF. From the sender side, it estimates the arrival time at the receiver of the packets to be sent. Then it tries to send in such a way that the packets arrive in order and in the shortest possible time at the receiver [10].

E. Block Estimation (BLEST)

It aims to reduce the HoL-blocking and thus potentially reduce spurious retransmissions. It estimates if an SF is going to cause HoL blocking by monitoring the send window and RTT, and adapts a scheduling policy to prevent such blocking. It mainly pauses sending data packets to avoid HoL blocking at the receiver. It waits for the faster SF despite the availability of CWND in the slower SFs to prevent HoL blocking [11].

III. Simulation Scenario

For the simulation testbed, we started by compiling NS3-DCE [7] in Ubuntu 16.04 LTS [14] kernel in VirtualBox [15]. We modified the NS3-DCE default installation to install the NS-3.33 instead of the default NS-3.34. Then we replaced the contents NS-3.33 folder with the NS-3.33 mmWave module [8], modified some files for NS3-DCE compatibility, then compiled and configured it to run with NS3-DCE. For the Linux Kernel integration, we changed the NS3-DCE configuration so that it downloads net-next-libos-4.4.0 with MPTCP Linux Kernel v0.90 [9], which is the latest MPTCP Linux Kernel compatible with NS3-DCE. We modified the Redundant, ECF, and BLEST scheduler's Linux implementation file from the newer version of the MPTCP Linux kernels to make them compatible with MPTCP Linux Kernel v0.90. Then compiled the kernel and configured it to run with NS3-DCE. Finally, we configured NS3-DCE itself so that it recognizes the changes.

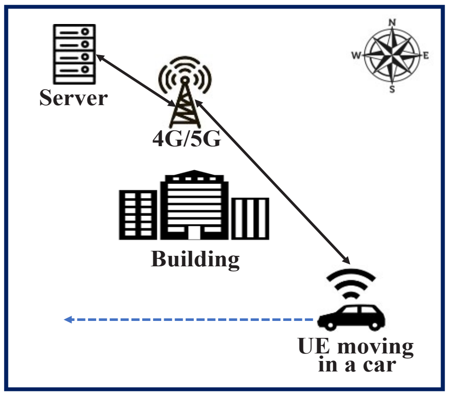

Fig. 2. The considered simulation scenario.

After preparing the simulation testbed, we designed a simulation scenario where 5G and 4G base stations are located at the same tower as shown in Fig. 2. The UE is in a car moving from the southeast corner to the southwest corner at a 60 km/h speed. At the start, there is a line-of-sight connection with the eNB and gNB. After some time, a building comes in between, causing non-line-of-sight connection. At the end, when the car passes the building, again, a line-of-sight connection establishes. In this scenario, we run two types of simulation experiments. In the first experiment, we start iperf data transfer at the beginning and continue for 20 seconds. In the second experiment, a fixed 10 Mb of data is sent via iperf continuously one after the other, from the start to the end of the experiment time of 20 seconds. We used the default Linux setup in both experiments except for the schedulers.

TABLE I. Performance of the Considered MPTCP Schedulers in terms of Total Sent Data and Flow Completion Time (FCT).

		SRTT	Round-robin	Redundant	ECF	BLESET
Total sent (*Mb*)	**5G**	479.1	479.9	464.9	479.8	487.7
	4G	283.5	277.8	471.2	221.2	641.8
FCT (*ms*)		27.2	28.4	22.4	31.1	19.7

IV. Performance Evaluation

For the first experiment, we observe the received signal to interference and noise ratio (SINR) and throughput by the UE. As shown in Fig. 3, as the UE moves, even during the non-line-of-sight connection, the SINR for the 4G network does not change much, as the 4G waves can easily penetrate through the house. Following the received SINR, the throughput does not fluctuate much for the SF using the 4G interface. However, the 5G network's SINR fluctuates a lot as it cannot easily penetrate the house. Following the losses in SINR, the throughput in the SF using the 5G connection also drops rapidly. Interestingly, although there is no change in the 4G SINR, the throughput of the SF going via 4G interface faces some losses in throughput. If we observe closely, this can be related to the high packet losses in the 5G SF. The severe losses in the 5G SF cause HoL blocking at the receiver, causing the 4G SF to stall, resulting in a decreased throughput.

Table 1 shows the total sent data through each SFs for the same experiment. The BLEST scheduler performs the best among the considered schedulers as it could successfully

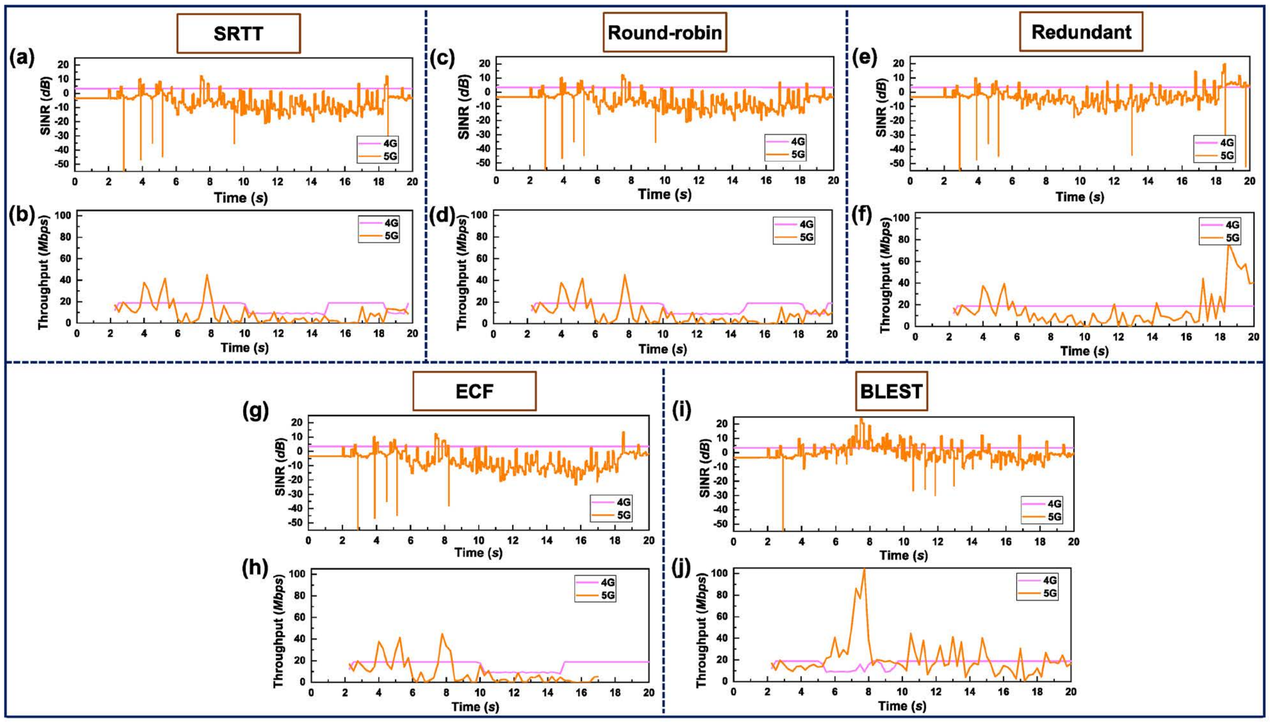

Fig. 3. Performance of the considered MPTCP schedulers: (a) received SINR and (b) throughput for SRTT, (c) received SINR and (d) throughput for round-robin, (e) received SINR and (f) throughput for redundant, (g) received SINR and (h) throughput for ECF, and (i) received SINR and (j) throughput for BLEST.

avoid the HoL blocking. The Redundant scheduler performs the second-best as it sends the same data through both the SFs to avoid HoL blocking. Interestingly, despite sending the same data through both the SFs, the Redundant scheduler's performance is lower than BLEST because the data does not arrive at the same time at the receiver due to the difference in RTT. And the ECF scheduler performs the worst as it stops sending through the 5G SF after some time. We believe that the ECF scheduler fails to correctly estimate the packet's arrival time at the receiver from the sender side because of the continuously changing throughput in the SF going through the 5G network. This failure causes a fault in ECF and leads it to stop the SF through the 5G network.

Table 1 also shows the second experiment's average FCT. As mentioned in the previous section, in this experiment, a fixed 10 Mb data is sent repeatedly throughout the 20 seconds experiment time. We measured the FCT for each transferred data and calculated the average as shown in Table 1. Similar to the previous experiment, BLEST takes the lowest FCT compared to the others. And ECF takes the highest FCT. It is worth mentioning that when ECF stops the SF going through the 5G interface, its FCT reduces significantly. Otherwise, the FCT could be much higher.

V. Conclusion

We investigated the behavior of five MPTCP schedulers in a scenario where the mobile UE tries to utilize both the 5G and 4G networks simultaneously. The results show that the BLEST scheduler performs the best among the considered MPTCP schedulers. We also found that the HoL blocking is the crucial factor that decides the actual performance of the schedulers.

In future work, we will further investigate the behavior of the different MPTCP schedulers in combination with other MPTCP congestion control algorithms.

Acknowledgment

This research was supported in part by the Basic Science Research Program through the National Research Foundation of Korea (NRF), funded by the Ministry of Education (No. NRF-2018R1A6A1A03025109), and by the National Research Foundation of Korea (NRF) grant funded by the Korean government (No. NRF-2019R1A2C1006249).

References

[1] L. Ding, Y. Tian, T. Liu, Z. Wei, and X. Zhang, "Understanding commercial 5G and its implications to (Multipath) TCP," *Computer Networks,* vol. 198, p. 108401, 2021.

[2] M. Series, "Minimum requirements related to technical performance for IMT-2020 radio interface(s)," Radiocommunication Sector of ITU (ITU-R)2017.

[3] L. Li *et al.*, "A longitudinal measurement study of TCP performance and behavior in 3G/4G networks over high speed rails," *IEEE/ACM transactions on networking,* vol. 25, no. 4, pp. 2195-2208, 2017.

[4] H. Ekström, "Non-Standalone and Standalone: Two Standards-Based Paths to 5G," ed: Ericsson, 2019.

[5] D. P. Malladi. (2019, 03 AUG 2022). *Key breakthroughs to drive a fast and smooth transition to 5G standalone*. Available: https://www.qualcomm.com/news/onq/2019/08/key-breakthroughs-drive-fast-and-smooth-transition-5g-standalone

[6] S. Ahmadi, *5G NR: Architecture, technology, implementation, and operation of 3GPP new radio standards*. Academic Press, 2019.

[7] H. Tazaki *et al.*, "Direct code execution: Revisiting library os architecture for reproducible network experiments," in *Proceedings of the ninth ACM conference on Emerging networking experiments and technologies*, 2013, pp. 217-228.

[8] M. Mezzavilla *et al.*, "End-to-end simulation of 5G mmWave networks," vol. 20, no. 3, pp. 2237-2263, 2018.

[9] R. Lübben and J. Morgenroth, "An odd couple: Loss-based congestion control and minimum RTT scheduling in MPTCP," in

2019 IEEE 44th Conference on Local Computer Networks (LCN), 2019, pp. 300-307: IEEE.

[10] Y.-s. Lim, E. M. Nahum, D. Towsley, and R. J. Gibbens, "ECF: An MPTCP path scheduler to manage heterogeneous paths," in *Proceedings of the 13th international conference on emerging networking experiments and technologies*, 2017, pp. 147-159.

[11] S. Ferlin, Ö. Alay, O. Mehani, and R. Boreli, "BLEST: Blocking estimation-based MPTCP scheduler for heterogeneous networks," in *2016 IFIP Networking Conference (IFIP Networking) and Workshops*, 2016, pp. 431-439: IEEE.

[12] B. Arzani, A. Gurney, S. Cheng, R. Guerin, and B. T. Loo, "Impact of path characteristics and scheduling policies on MPTCP performance," in *2014 28th International Conference on Advanced Information Networking and Applications Workshops*, 2014, pp. 743-748: IEEE.

[13] B. Felix, I. Steuck, A. Santos, S. Secci, and M. Nogueira, "Redundant packet scheduling by uncorrelated paths in heterogeneous wireless networks," in *2018 IEEE Symposium on Computers and Communications (ISCC)*, 2018, pp. 00498-00503: IEEE.

[14] R. Khalili, N. Gast, and M. Popovic, "Opportunistic linked-increases congestion control algorithm for MPTCP," BCP 78-79, 2013.

[15] F. Kelly and T. Voice. (2005) Stability of end-to-end algorithms for joint routing and rate control. *ACM SIGCOMM Computer Communication Review*. 5-12.